# COMPUTER LANGUAGE™

$2.95
Canada $3.95

VOLUME 2, NUMBER 5
MAY 1985

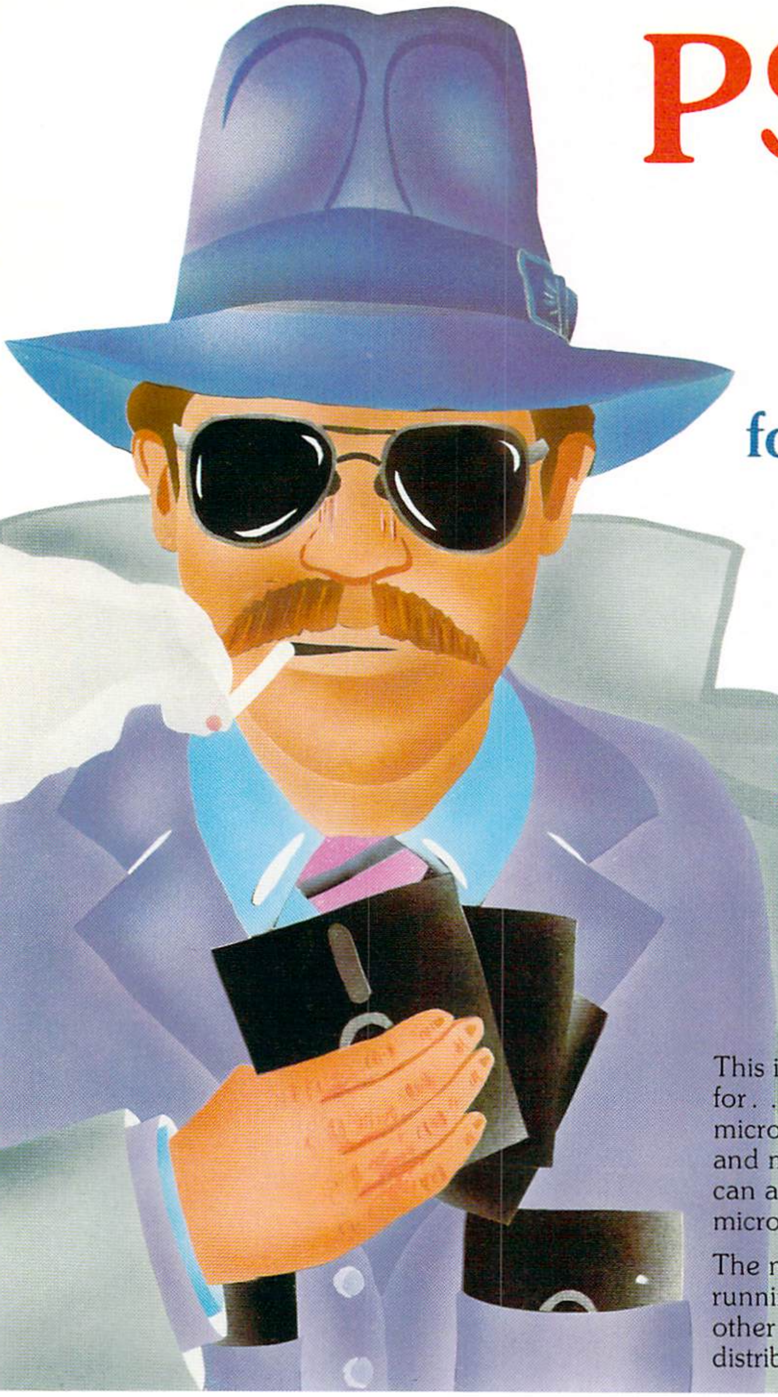
MACROS IN BASIC

AN OBJECT-ORIENTED LANGUAGE CALLED CLASCAL

TWENTY-NINE BASICS COMPARED

[redacted]

EXCLUSIVE INTERVIEWS:
NIKLAUS WIRTH AND DONALD KNUTH



# PSSST...

# JANUS/Ada

# for $99.95!!

*PRESENTING THE NEW JANUS/Ada C-PAK!!*

1. **Janus/Ada Compiler**
2. **Janus/Ada Linker**
3. **Janus/Ada Libraries**
4. **Janus/Ada Example/ Programs**
5. **Janus/Ada User Manual**

*AND THESE ADDED FEATURES!!*

1. **Free User's Group**
2. ***$99.95 Discount* on the Janus/Ada D-Pak**
3. **No License!!**
4. **No Copy Protection!!!**
5. **Customer tested for over 3 years!!!**

This is the introductory Ada™ package you've been waiting for . . . over three years of actual field use, specifically on microcomputers, by the government, Fortune 500 businesses and major universities. Realistically priced, at $99.95, so *you* can afford the most popular Ada implementation used on microcomputers!

The new "C"-Pak is available for most microcomputers running MS-DOS, including the IBM PC AT™, as are all the other fine Janus/Ada programs. Call us or an authorized distributor for your copy today!

National Distributors

Westico, Inc.
25 Van Zant St.
Norwalk, CT 06855
(203) 853-6880

ASH II
7407 Marisol
Houston, TX 77083
(713) 933-1828

A.O.K. Computers
816 Easley St., Suite 615
Silver Springs, MD 20910
(301) 588-8446

Trinity Solutions
5340 Thornwood Dr., Suite 102
San Jose, CA 95123
(408) 226-0170

MicroProgramming, Inc.
P.O. Box 3356
Chatsworth, CA 91313
(818) 993-6475

International Distributors

Ada Australia
218 Lutwyche Rd.
Windsor 4030
QLD. Australia
(07) 57 9997

Progesco
155, rue du Faubourg
St. -Denis
75010 Paris
France
(1) 205.39. 47

Lifeboat, Inc. Japan
3-6, Kando-Nishikicho
Chiyoda-ku
Tokyo 101, JAPAN
03-293-4711

CP/M, CP/M-86, CCP/M-86 are trademarks of Digital Research, Inc.
*ADA is a trademark of the U.S. Department of Defense
MS-DOS is a trademark of Microsoft

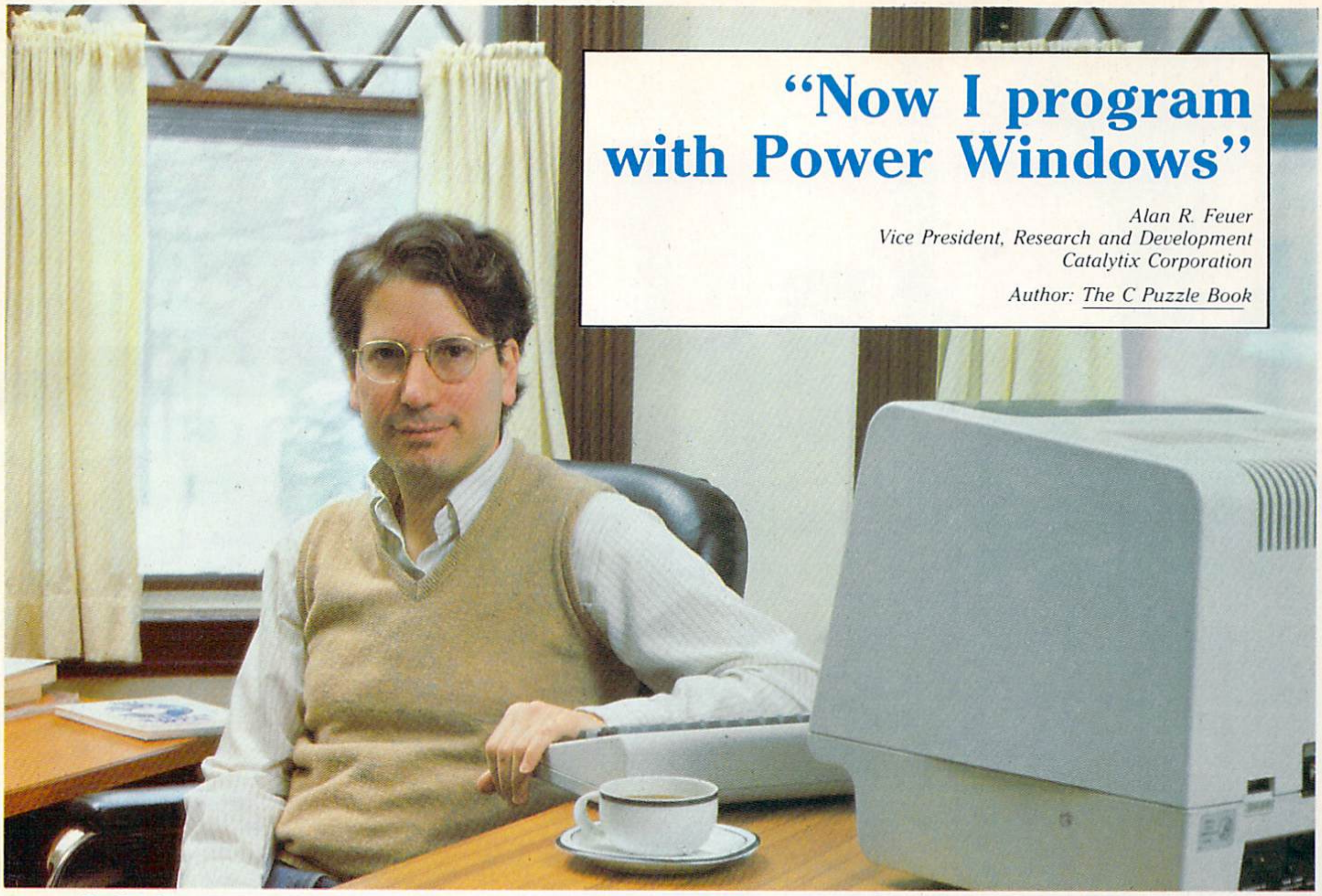
©Copyright 1984 RR Software

RR SOFTWARE, INC.

*specialists in state of the art programming*

P.O. Box 1512 Madison, Wisconsin 53701
(608) 244-6436 TELEX 4998168

CIRCLE 58 ON READER SERVICE CARD



## "Now I program with Power Windows"

*Alan R. Feuer*
*Vice President, Research and Development*
*Catalytix Corporation*

*Author: The C Puzzle Book*

# CCA EMACS...The Most Powerful Editor Environment Available for Unix and VAX/VMS

*"Programming with CCA EMACS, I can look at two or more files at once in different windows and then move text between them."*

Alan Feuer is just one of many demanding programmers who have discovered that CCA EMACS™ makes program editing and system development much easier and faster. And "power windows" are only part of the reason Alan Feuer uses CCA EMACS. . .

Unprecedented power, speed, functionality, extensibility, pliability, and consistency across systems and on *any* terminal are others. CCA EMACS includes close to 400 built-in commands which let you do any job with only a few keystrokes, even the kinds of things that are difficult or impossible with other editors. And with our full Common Lisp-based extension language, Elisp™, you can customize CCA EMACS to meet all your specific program needs.

CCA EMACS has two extensive recovery facilities to protect against system failures. Supported by a full online documentation package, including tutorial, the system can be used by beginners and experts alike.

This complete kit of editing tools runs under Berkeley Unix™ (4.1BSD and 4.2BSD), Bell Unix (Systems III and V), Xenix™, and VAX/VMS™.
Binary prices range from $380 to $850 for Unix to $1900 for VMS.

**CCA Uniworks, Inc.**
*Productivity Tools for Programmers*
20 William Street, Wellesley MA 02181

**CIRCLE 61 ON READER SERVICE CARD**

For more information or to place an order call our customer representatives at

**800-222-0214**

in MA (617) 235-2600
or mail this request form today.

Please send me information on:
☐ CCA EMACS
☐ AI Development Tools
☐ The Safe C Development Tools
☐ Your complete line of state-of-the-art programming tools

☐ Please send license forms

Name ____________________
Title ____________________
Company ____________________
Address ____________________
City, State, Zip ____________________
Phone (______) ____________________

**CCA UNIWORKS, INC.**
20 William Street Wellesley, MA 02181

A Crowntek Company

Unix, VAX and VMS and Xenix are trademarks of Bell Laboratories, Digital Equipment Corporation, and Microsoft Corporation, respectively, Safe C is a trademark of Catalytix Corporation, CCA EMACS and Elisp are trademarks of CCA Uniworks, Inc. CL585

*Clipper gives dBASE III™ users more time to do more. Or less.*

Clipper™ allows you to run all dBASE III™ programs 2 to 20 times faster than they do with the standard dBASE interpreter.

That frees up extra time you're wasting if you're running dBASE III programs without Clipper.

Extra time to think. To create. To produce. To use as you choose.

You see, Clipper is the first true compiler for dBASE III. Clipper eliminates the time-consuming translation which the dBASE interpreter performs line after line whenever a program is run.

With Clipper, once you've debugged your source code, it's compiled into more efficient machine code.

And Clipper compiles all your dBASE III programs. The ones you have today. The ones you'll have tomorrow. But don't wait until tomorrow to order Clipper.

Today, Clipper has already been purchased to speed up dBASE run time at 3M and Touche Ross. At Exxon and NASA. In the Harvard Physics Department. For the State of Arizona and TRW.

And that's just a few of the installations worldwide. From Greece to Venezuela to Canada to Europe.

So stop wasting time. Call our toll-free 800 number and get Clipper.

You'll spend less time running dBASE III and more time running the rest of your life.

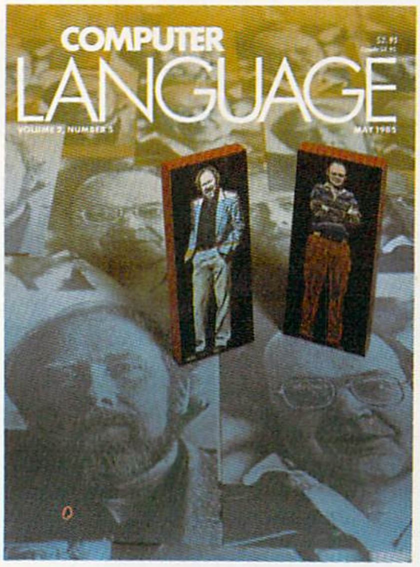
Nantucket™

5995 S. Sepulveda Blvd., Culver City, CA 90230 (800) 556-1234 ext. 225 In California (800) 441-2345 ext. 225

**CIRCLE 59 ON READER SERVICE CARD**

dBASE III is a registered trademark of Ashton Tate.

# COMPUTER LANGUAGE



## ARTICLES

**Programming Philosophy: Interviews with Donald Knuth and Niklaus Wirth** **25**

*by Ken Takara*

Renowned computer scientists Donald Knuth and Niklaus Wirth cover a variety of historical, technical, and philosophical issues in interviews with *COMPUTER LANGUAGE*'s Designers Debate columnist. Specific topics discussed range from computer science as an art to the merits of structured programming. Knuth and Wirth are both recipients of the Association for Computing Machinery's prestigious Turing Award. Knuth, who received the award in 1974, is particularly well-known for his three volumes of *The Art of Computer Programming*. Wirth received the Turing Award in 1984 for his development the Euler, ALGOL-W, Pascal, and Modula computer languages.

**Macros in BASIC** **37**

*by Dwaine L. Bendorf*

If you program in BASIC and are willing to modify your program development cycle to include the use of a structured preprocessor, you may benefit from the use of predefined macros. Since code development in BASIC is often repetitive, programming with macros can give you the ability, for example, to save and reuse complicated algorithms. The preprocessor discussed in this article has been put into the public domain by the author.

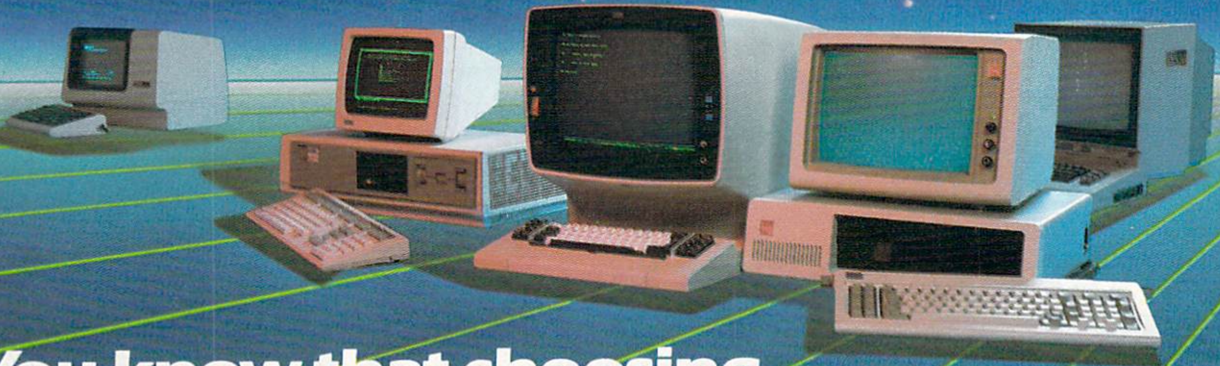
**BASIC Recursion** **43**

*by Hugh Aguilar*

Using a familiar puzzle called the Towers of Hanoi, this article explores how recursion in BASIC can be much more than just a procedure that calls on itself as a subprocedure. Solving this puzzle will help the reader understand how the process of recursion allows for the reduction of a problem into simpler and simpler subsets of itself until solutions can be found.

## DEPARTMENTS

**Editor's Notes** ______ **5**
**Feedback** ______ **7**
**CrossThoughts** ______ **13**
Dynamic hashing in external searching
**Public Domain Software Review** ______ **21**
Macintosh and MacBASIC
**Exotic Language of the Month Club** ______ **49**
Clascal—An object-oriented Pascal
**Product BINGO** ______ **55**
**Software Review** ______ **57**
Roundup of 29 BASIC interpreters and compilers
**Advertiser Index** ______ **96**



## You know that choosing the right software is serious business. So does WATCOM.

So before you make any decisions about your software needs, talk to WATCOM— the people major software users around the world have trusted for years.

WATCOM has the products you need to get the job done right. Proven performers like WATFOR*, WATFIV*, WATBOL*, and SCRIPT. Plus new leaders in software for PC workstations and micro-to-mainframe communications. Networks, language interpreters and compilers. Text preparation and data management.

All WATCOM products are human engineered to provide the optimum in people efficiency and productivity. And they're designed to run compatibly on IBM mainframes and PC's, Digital mainframes and micros, and Commodore micros.

Whatever you need is backed up by WATCOM's innovative maintenance and support services. You'll be kept up to date with the latest in product enhancements and information. And our publications and seminars will help you get the most out of your software investment.

WATCOM. Quality products. Professional service. And a reputation built on more than 150,000 licensed mainframe and micro software programs throughout the world. So talk to us before you decide. After all, choosing the right software is serious business. For you. And for WATCOM.

## Make the right choice: *WATCOM INTERPRETERS*

Excellent error diagnostics make WATCOM Interpreters the right choice in software for efficient program development in APL, BASIC, COBOL, FORTRAN, or Pascal.

WATCOM Interpreters emphasize error detection so that program corrections are more easily executed. Hard-to-find errors can be quickly located with the integrated debugging system for COBOL, FORTRAN, and Pascal. And programs can be efficiently entered and corrected with the integrated full-screen editor in all languages but APL.

WATCOM Interpreters are available for IBM PC, IBM 370 VM/SP CMS, and Digital VAX VMS*.

Make the right choice. Call or write WATCOM today and we'll tell you all about WATCOM Interpreters or any of WATCOM'S other people-efficient products.

WATCOM

**The right choice in software.**

Yes! I want to make the right choice in software. Send me more information on: ☐ WATCOM INTERPRETERS ☐ WATCOM Software Catalogue

Name:____________________

Company:____________________

Title:____________________

Address:____________________

City:__________ State:__________ Zip:__________

**WATCOM PRODUCTS INC.**
**415 Phillip Street**
**Waterloo, Ontario, Canada**
**N2L 3X2**
(519) 886-3700
**Telex 06-955458**

*WATFOR, WATFIV and WATBOL are registered trademarks of the University of Waterloo.
*IBM PC and IBM 370 VM/SP CMS are registered trademarks of International Business Machines Corporation.
*VAX, VMS are registered trademarks of Digital Equipment Corporation.

**CIRCLE 28 ON READER SERVICE CARD**

# Editor's Notes

Is programming an art or a science, or both? Do structured languages enforce good programming style? What are the most elegant ways to make your code reflect its underlying algorithms?

Who better to ask than the world's two most well-known computer scientists—Donald Knuth and Niklaus Wirth. In two thought-provoking interviews, Knuth and Wirth discuss many historical, technical, and philosophical issues of interest to programmers from all language backgrounds.

Donald Knuth, a professor at Stanford University, Palo Alto, Calif., is renowned for his three volumes of *The Art of Computer Programming*, which is considered a landmark in formalizing the field of computer science. He was the 1974 recipient of the Association for Computing Machinery's prestigious A.M. Turing Award. Knuth has recently become involved in applying computer science and mathematics to typeface design and typesetting.

Niklaus Wirth, currently on sabbatical at the Xerox Palo Alto Research Center, Palo Alto, Calif, is the most recent recipient of the ACM Turing Award. He is best known for his creation of Pascal and Modula—highly structured and strictly typed languages—and for his strong advocacy of structured programming.

*COMPUTER LANGUAGE*'s Designers Debate columnist, Ken Takara, visited with Knuth and Wirth in their Silicon Valley offices and discovered some interesting differences of opinion each has on certain contemporary programming topics. Ken's column will return next month.

And that's not all that's special about this issue!

This month features a comprehensive analysis of 29 BASIC interpreters and compilers for the IBM, CP/M, and Macintosh computers. From the surprising amount of product development in the BASIC sector—especially implementations written for the IBM PC—it's apparent that the BASIC language is still going strong.

And, not unlike the special three-part series on programming with macros in C (which began with the February C issue), this month we present an article that will show you how to use a structured preprocessor to get the same kind of macro programming power provided by the C language.

Also this month is an article on recursion in BASIC. Using the Towers of Hanoi puzzle, the author demonstrates how recursion allows for the reduction of a problem into simpler and simpler subsets of itself until a solution can be found.

Watch for a special announcement next month!

Craig LaGrow

**Craig LaGrow**
**Editor**

### Telecommunicate to COMPUTER LANGUAGE

*COMPUTER LANGUAGE* has established two bulletin board systems for you to upload and download text and binary programs, as well as to leave your own electronic Letter to the Editor. All the program listings referred to in every issue of the magazine will be available here.

For those readers without access to a modem who desire a copy of program listings referred to but not printed in an issue, send $5 to *COMPUTER LANGUAGE*, attention: Listings Dept., 131 Townsend St., San Francisco, Calif. 94107. We will mail you a copy of all the listings not printed in this issue.

In addition, *COMPUTER LANGUAGE* has its own Special Interest Group on CompuServe's national data base. After calling into your local CompuServe node, simply type "GO CLM" at any prompt and you'll be in our SIG.

To access our bulletin board, set your computer or terminal to the following parameters: 8 data bits, no parity, 1 stop bit, full duplex, and either 300 or 1200 baud. The telephone number is (415) 957-9370. After your modem makes the connection, type RETURN several times, and everything else is easy.

Both systems are open 24 hours per day, 7 days per week. Due to the heavy number of callers, please do not log into the system more than one time per day. Messages left on either system will be combined the following day.

COMPUTER
LANGUAGE

**EDITOR**
Craig LaGrow

**MANAGING EDITOR**
Regina Starr Ridley

**TECHNICAL EDITOR**
John Halamka

**PRODUCT REVIEW EDITOR**
Hugh Byrne

**EDITORIAL ASSISTANTS**
Lorilee Biernacki, John Harrington

**CONTRIBUTING EDITORS**
Tim Endres, Doug Millison, Tim Parker, Namir Clement Shammas

**INDUSTRY NEWS CONSULTANT**
Bruce Lynch

**SPECIAL PROJECTS MANAGER**
Jan Dente

**OPERATIONS CONSULTANT**
Beatrice C. Blatteis

**CIRCULATION COORDINATOR**
Renato Sunico

**ART DIRECTOR**
Jeanne Schacht

**COVER PHOTO**
Dow Clement Photography

**PRODUCTION ARTIST**
Anne Doering

**PRODUCTION**
Barbara Luck, Steve Campbell, Kyle Houbolt

**TECHNICAL CONSULTANT**
Addison Sims

**ACCOUNTING MANAGER**
Lauren Kalkstein

**WHOLESALE COORDINATOR**
Nicola Sullivan

**PUBLISHER**
Carl Landau

COMPUTER LANGUAGE *is published monthly by* COMPUTER LANGUAGE *Publishing Ltd., 131 Townsend St., San Francisco, CA 94107. (415) 957-9353.*

*Advertising: For information on ad rates, deadlines, and placement, contact Carl Landau at (415) 957-9353, or write to:* COMPUTER LANGUAGE, *131 Townsend St., San Francisco, CA 94107.*

*Editorial: Please address all letters and inquiries to: Craig LaGrow, Editor,* COMPUTER LANGUAGE, *131 Townsend St., San Francisco, CA 94107.*

*Subscriptions: Contact* COMPUTER LANGUAGE, *Subscriptions Dept., 2443 Fillmore St., Suite 346, San Francisco, CA 94115. Single copy price: $2.95. Subscription prices: $24.95 per year (U.S.); $30.95 per year (Canada and Mexico). Subscription prices for outside the U.S., Canada, and Mexico: $36.95 (surface mail), $54.95 (air mail)—U.S. currency only. Please allow six weeks for new subscription service to begin.*

*Postal information: Second-class postage rate is pending at San Francisco, CA and additional mailing offices.*

*Reprints: Copyright 1985 by* COMPUTER LANGUAGE *Publishing Ltd. All rights reserved. Reproduction of material appearing in* COMPUTER LANGUAGE *is forbidden without written permission.*

*Change of address: Please allow six weeks for change of address to take effect. POSTMASTER: Send change of address (Form 3579) to* COMPUTER LANGUAGE, *131 Townsend St., San Francisco, CA 94107.*

COMPUTER LANGUAGE *is a registered trademark owned by the magazine's parent company, CL Publications. All material published in* COMPUTER LANGUAGE *is copyrighted © 1985 by CL Publications, Inc. All rights reserved.*

# Potent Pascal.

Microsoft® Pascal may be the most powerful software development environment available for the MS™-DOS system. It combines the programming advantages of a structured high-level language with the fast execution speed of native code compilation.

And it exceeds the proposed ISO and ANSI standards with logical extensions that make the language more powerful and versatile. For example, programming capabilities even allow you to manipulate data at the system and machine level.

It gives you single and double precision IEEE floating point arithmetic. Numeric operations take advantage of the 8087. Or automatic software emulation is provided if the coprocessor is not installed.

**MICROSOFT®**
The High Performance Software

Support for long heap allocation and separate module compilation gives you the flexibility to create large programs up to one megabyte.

And the standard linking interface makes it easy to combine Microsoft FORTRAN or assembly language subroutines.

**Call 800-426-9400 to order the potent Pascal. $300.***

In Washington State, call 206-828-8088. Ask for operator K5, who will rush you your order, send you more information, or give you the name of your nearest dealer to see Microsoft Pascal in action.

*Price exclusive of handling and Washington State sales tax.
Microsoft is a registered trademark and MS is a trademark of Microsoft Corporation.

CIRCLE 43 ON READER SERVICE CARD

# FEEDBACK

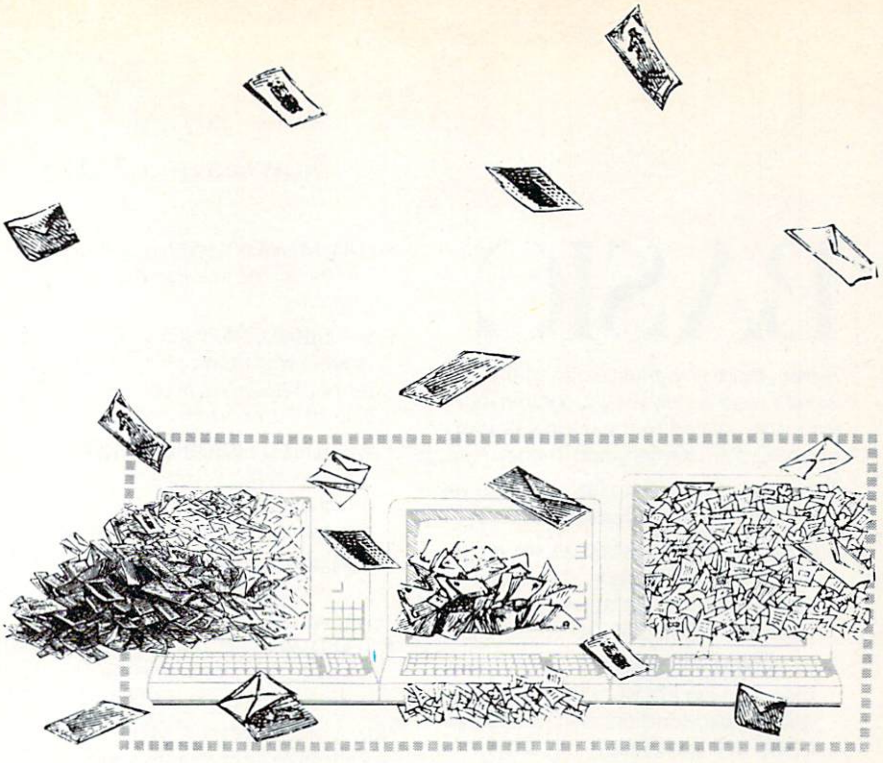


Illustration: Anne Doering

## Forth is it

Dear Editor:

I enjoyed your March Designers Debate involving Charles Moore and the other eloquent proponents of the Forth language. Such a roundtable discussion on the language does a great deal to explain what Forth is all about and why we users (the so-called fanatics) are so in love with it.

Moore single-handedly invented Forth—a fast, extensible, interactive, and structured programming language—to dramatically improve a programmer's productivity. That his invention has worked spectacularly well is evidenced by the tens of thousands of Forth programmers who have adopted it out of desperation because no other language provides such flexibility, speed, compactness, and ease of use.

It has often been said that Forth will be displaced by LISP or C. I doubt that very much because unlike any other language, Forth addresses the fundamental issue of what a computer language should be: an efficient and productive compromise between a human being and a computer.

*George D. Dooley*
*State College, Penn.*

## Don't turn them loose

Dear Editor:

Any doubts in the world at large that Forth belongs in its own little subculture and should not be turned loose on real projects should be effectively dispelled by March's Designers Debate. If *COMPUTER LANGUAGE* expects its readers to be conversant in more than one language, I don't think it unreasonable to expect the same of its panelists.

I do have an encouraging word for Gregg Parsons, the lone infidel at your little prayer breakfast (Debate? Ha!). There are indeed real live professional programmers who sometimes use Forth. However, you'll seldom see us at FIG meetings, because we don't always use Forth. We're not particularly interested in lectures on how to sharpen the Forth screwdriver so it can cut wood almost half as fast as the C saw or exhortations to throw down our PL/I hammers and fasten everything with screws. We're also tired of the endless debate over whether the amateur woodworkers will choose Phillips or Torx to make our vast inventory of slotted screws obsolete.

*Ran Talbott*
*Los Gatos, Calif.*

## Forth R.I.P.

Dear Editor:

A science museum is considering designing a custom robot for use in an educational summer camp. A team consisting of a programmer, an educator, a machinist, and a controls technician (me) discusses how such a thing might be made practical. The museum takes a very intuitive approach to software and many teachers will be modifying the software to suit their special needs. A member of the robotics society thinks Forth is easily the best language in which to write our LOGO-like interpreter.

The 75% of the team who are not programmers and teachers who will later be modifying the code need some guideposts for communicating with the programmer. These might be Bryant-Kitch diagrams, pseudocode, and/or Warnier-Orr diagrams. The correspondence between these aids and the actual code will be closest if a structured language with Pascal-like procedure calls is used. This argues against Forth but not fatally.

If the museum wishes to sell the object code to another educational institution, Forth requires that its kernel and its license go with the object code. Again just annoying, not fatal.

When it comes to readability, the usual complaint is that Forth's arithmetic is expressed in postfix notation (which is actually much easier to learn than assembler mnemonics). But a 40-line routine in Forth will likely contain 40 new key words, the meanings of which must be learned before the routine can be understood.

If a teacher encounters an unfamiliar key word in a C routine, once understood, the word will be understood in any other C program the teacher encounters and will most probably correspond to reserved words found in Ada, ALGOL, Augusta, Modula-2, Pascal, PL/M, RATFOR, Tiny-HI, and other structured languages. Not so in Forth. Each time a new Forth program must be read, a whole new vocabulary must be learned. As our code will undergo continuous evolution, I suspect that will be fatal to the use of Forth in our application.

*Steven S. Coles*
*Seattle, Wash.*

## More is not less

Dear Editor:

I am writing this in response to Namir Shammas's CrossThoughts column.

Mr. Shammas, you have been sitting under the pyramid too long. I, for one, am weary of a so-called profession going about reinventing the wheel. In my personal opinion, programming language development would be more worthwhile if it were absolutely Wirth-less. Ever since

The BASIC idea.

Now it's even better.

*True* BASIC™

A new, more powerful version of the world's most widely used programming language–created by the original developers of BASIC, John Kemeny and Thomas Kurtz.

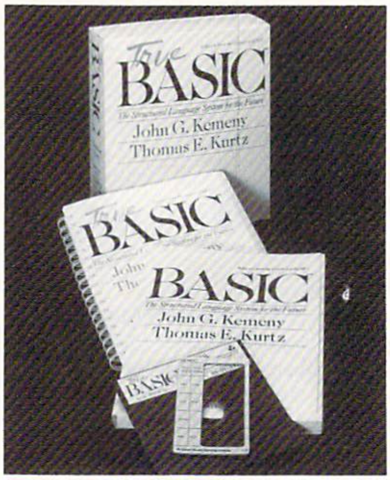
True BASIC is still easy to learn and use yet offers the following advanced features:

- **STRUCTURE**–True BASIC allows you to write modular programs. It supports advanced control structures such as SELECT CASE, IF-THEN-ELSEIF, DO Loops, etc.
- **EXTERNAL PROGRAMS**–Increase your programming efficiency by storing frequently used functions and subroutines in user-defined libraries. True BASIC supports calls to assembly language subroutines.
- **FULL MEMORY**–With True BASIC you can use all the available memory in your computer.
- **SUPERIOR GRAPHICS**–Allow you to draw in your own coordinates, not in terms of pixels. Animation, sound and color are supported.
- **IMPROVED ERROR CHECKING**–Compiler reports syntax errors before running a program. The language incorporates user-defined error messages.
- **WINDOWS**–True BASIC allows you to program multiple screen windows, each with its own parameters.
- **SPEED**–Because True BASIC is compiled, it is faster than interpreted BASICs. Automatic 8087 support is standard.

Now available for the IBM PC. Apple Macintosh available Summer, 1985.

True BASIC is a trademark of True BASIC, Inc./IBM is a registered trademark of International Business Machines, Inc./Apple is a trademark of Apple Computer, Inc./Macintosh is a trademark licensed to Apple Computer, Inc.

CIRCLE 7 ON READER SERVICE CARD



True BASIC.™
The BASIC idea made better.

Addison-Wesley
Reading, Massachusetts, 01867
617/944-6795

---

Your PC can now "speak" C when using...

BASTOC™

A BASIC-To-C Translator

**JMI's BASTOC is a versatile software tool which converts BASIC source programs to C source programs. BASTOC also translates multiple dialects of BASIC.**

Registered Trademarks: IBM—International Business Machines Corp., Microsoft, MS, XENIX—Microsoft Corp.; Applesoft—Apple Computer, Inc.; TRS-80—Tandy Radio Shack Corp.; CBASIC—Digital Research, Inc.

Distributors: **Australia**, Fawnray Pty. Ltd., P.O.B. 224, Hurtsville, NSW 2220 (612) 570-6100; **Japan**, Advanced Data Controls, Corp., Chiyoda-ku, Tokyo (03) 263-0383; **United Kingdom**, Real Time Systems, Newcastle upon Tyne, 0632-733131

**BASTOC FEATURES**

- Supports Microsoft BASIC or DRI CBASIC.
- Full run-time support library is provided.
- Provides conventional BASIC compiler when used in conjunction with a C compiler.
- Produces formatted and structured C code, easily maintained and modified.
- The output of BASTOC may be used directly as input to a standard C compiler.
- Several dialects of BASIC may be translated to C on the same system.
- BASTOC will soon translate additional dialects, including SMC BASIC.

**AVAILABLE NOW**

JMI's BASTOC is available now for the IBM PC and PC-compatible computers using PC-DOS or MS-DOS, and the Radio Shack TRS-80, Model 16, running XENIX. BASTOC will be available for other systems soon!

**ORDER NOW**

The BASTOC single unit binary price is $350, and includes documentation, media, and shipping. To order, or to get more information, write or call JMI Software Consultants, Inc., 215-628-0846. Check, Money Orders, VISA and MC are acceptable.

jmi

**JMI SOFTWARE CONSULTANTS, INC.**
**P.O. BOX 481 • 904 SHEBLE LANE**
**SPRING HOUSE, PA 19477 • 215-628-0846**

CIRCLE 65 ON READER SERVICE CARD

John Kemeny and Thomas Kurtz foisted their BASIC failure on students who were unable to protect themselves, we have experienced one mess after another in so-called programming languages.

Literature in the 1960s and 1970s spawned much stream of consciousness junk because people were encouraged to do their own thing. They were free to ignore all that had gone on before, because there was no sense in being bound by old rules.

From the universities, from the bowels of computer science, sprang a new set of heroes. Their accomplishments were in accord with what could be easily achieved. That is, if something were difficult, it could be ignored (and dismissed) as either unnecessary or outside the scope of effort. In the case of one man, the architecture he found was unsuited to his linguistics concepts. He has since gone on to gild his Lilith.

By not being bound by the old rules, they set about reducing our capabilities. When I set about learning under the new rules, I discovered I had some restrictions I didn't have before.

I found I had to define my data before I could use it. Previously it could appear anywhere in the block in which it was local. Thus I could have an action followed by a description if I thought this would increase the readability.

I used to be able to invoke a function and selectively change protect any or all arguments. That is, the compiler would create a run-time copy of any type or size of element or aggregate. I didn't find myself with different rules for elements and aggregates.

I could perform arithmetic, Boolean, and string operations on aggregates and aggregate cross-sections. I had strong typing with defined conversions between types. When I wanted to define my own conversion sequence, I had to explicitly do so, such that whoever came later would see what I had done and, more importantly, what I had intended be done. You would find no need for lint in my navel.

I used a language that had key words but no reserved words. This forced it to use statement terminators. Then again, everything was a statement. The program was a sequence of statements, not lines. That lines should appear was a restriction of the medium.

To the compiler, the program was a single continuous stream. I could define to the compiler that portion of the medium lines which was to be considered as part of the stream and that which was to be ignored. In this the medium became a freer form document.

Because the list could go on and on, I will cut it mercifully short with one last item. I, the programmer, could define the encoded form—width and precision—of arithmetic data. Thus I was never faced with the problems of portability of so-called modern programming languages.

Please understand when I cannot seriously believe your idea of proposing a subset of Ada. The last time the Dept. of Defense imposed its will on the profession we got COBOL. Not even modern programming languages are that retarded. If you must give it a name, call it CL/1. Then let's have an orderly progression from there.

For me, the candidates for consideration are LISP, Forth, APL, SNOBOL, and PL/I. No Pascal, Modula-2, C, or Ada. If we are to engage in this exercise, then in the the name of extension, let's end up with more and not less. 'Tis a Turing mind who believes that less is more. Orwell will be the 1985 winner.

LISP, Forth, APL, SNOBOL, and PL/I. I'll take that team against any and all comers. Let the games begin.

*Lynn H. Maxson*
*Oceanside, Calif.*

**Sieve of Eratosthenes test**

| Language | Machine | Time (sec, 10 iterations) |
|---|---|---|
| BASIC | T199/4A | 3,960 |
| ASSEMBLER | T199/4A | 7 |
| COBOL (RM) | TRS80-2 | 3,390 |
| BASIC | TRS80-2 | 1,610 Floating |
| BASIC | TRS80-2 | 1,210 Integer |
| Assembler | TRS80-2 | 14 |
| BASIC | IBM PC | 1,800 Floating |
| COBOL (CIS) | IBM PC | 1,700 |
| BASIC | IBM PC | 1,330 Integer |
| BASCOM | IBM PC | 350 Floating |
| COBOL (mbp) | IBM PC | 220 Integer |
| Forth | IBM PC | 70 |
| BASCOM | IBM PC | 41 Multi-Link |
| C | IBM PC | 23 C-Systems |
| Pascal | IBM PC | 16 TURBO |
| BASCOM | IBM PC | 16 Integer |
| COBOL (Realia) | IBM PC | 9 |
| Assembler | IBM PC | 4 |
| BASIC | TRS-2000 | 1,200 Floating |
| BASIC | TRS-2000 | 890 Integer |
| BASCOM | TRS-2000 | 4 Integer |
| COBOL | IBM 4331-2 | 14 |
| PL/I | IBM 4331-2 | 4 |
| Assembler | IBM 4331-2 | 1 |

Table 1.

## PPL thoughts

Dear Editor:

I wonder if I am reading things wrong, but there seems to be a conceptual error in Namir Shammas's March CrossThoughts. On page 16, listing 2, WHEN | block, shouldn't the indices [SH-1] and [SH] be swapped in the first four cases of the CASE Operation?

*Daleep Bhatia*
*Gaithersburg, Md.*

*Namir Shammas responds: No, because Drop stack signifies two operations: flopping the top two stack elements followed by popping the topmost one. This kind of confusion may be a side effect of using PPL and limited space—not all details can be included.*

## Enough evidence

Dear Editor:

I write this in response to Christopher Pettus's letter in the March issue responding to my letter in January.

Consider the Sieve of Eratosthenes. The merchants of complexity (academics) say it's too simpleminded. The purveyors of fear and uncertainty (salesmen) say it's meaningless. The losers say it's not representative. Considering the opposition, you know we must be on to something good.

Why does a TRS-80 (4MHz Z80) run BASIC faster than an IBM PC (Table 1)? Speed differences due to hardware are insignificant compared to software efficiency—10:1 vs. 200:1. So forget about clock speed, MIP rate and path width; worry about operating system compiler and programming environment. As an exercise, see if you can figure out which compilers produce p-code and which generate machine language.

COBOL is inherently faster than the others because of its richness of data types and verbs. The Sieve, for example, spends a lot of its time searching a table. Every language has arrays and subscripts, but only COBOL has a fast table lookup verb (*INSTR* works on strings, not tables, and it isn't fast). When I write SEARCH, Realia generates a "REP SCASB." There's no way a C compiler could generate that instruction because the C language doesn't have a search operation.

Before Realia, COBOL compilers were written by people who were C programmers at heart. They're like the car mechanic who works on Cadillacs but drives a BMW. Lacking a feel for the language, they did a sloppy job. The benchmark shows just how sloppy.

COBOL beats the number crunchers on the Sieve, it was used to write an outstanding compiler (Realia), and it's also good for "general ledgers, payrolls and stuff like that." What more evidence do you want?

*Robert Wagner*
*Lubbock, Texas*

## The price is right

Dear Editor:

In March's Industry Insight, Bruce Lynch decried the cost of current software for micros and suggested the price should be set by the cost to duplicate and provide documentation. For this to work, every package sold would require sales in the $100,000 range.

The cost to hire a good programmer and provide a work place today is a minimum of $50,000, with $100,000 common. This programmer can only produce 2,000 lines (or so the Yourdons of the world say) of code a year. So a 20,000-line package costs $5 million for the code alone. On the other hand, $500 for a word processor and $10,000 for a CAD system seem high. The problem with software costs is that there is little marketplace pressure to set prices, so most software is sold like audio products (with most being sold in outlets that are owned by audio stores).

The major point left out of the column was the license a user is forced to accept (whether you sign or not), and the contract many mail order houses use. Almost all licenses say you can't copy the software because it's copyrighted, but we don't have to make it work (As Is clause). Companies must cover their gross liabilities, but to say pay me $300 for a item that has less protection than a car is poor. The new key/keyring protection devices will help solve the theft problem, but fair prices and some support services would go farther.

*Larry Chandler*
*Eagan, Minn.*

## C/80 correction

*In our recent C compiler product wrap-up (*COMPUTER LANGUAGE, *February 1985), we made a few factual errors that concern the Toolworks C/80 compiler from The Software Toolworks. We now set the record straight:*

- *Toolworks C/80 is not "a late entry into the CP/M group of compilers." It has been sold since 1980.*
- *Toolworks C/80 is not "about $90." The total for the basic compiler and the float/long upgrade is $79.90.*
- *Toolworks C/80's failure to compile and run the deref benchmark was due to an obscure bug in the benchmark program. The compiler was not at fault.*
- *The conclusion that "The main effort (in writing a large program with C/80) would be to verify which constructs worked and which were going to go out to lunch" was based on the benchmark problem and is withdrawn.—Ed.*

You're in Good Company When You Program in BetterBASIC

JOHN
BOB'S SOFTWARE
IBM
TRW
GE
GEORGE
STAN
Mobil
AT&T
4

All of these companies rely on BetterBASIC to write their software programs. They have found that BetterBASIC combines the features they need from BASIC, Pascal, C and Forth in one familiar environment. Some of these features include the following.

**640K** Now you can use the full memory of your PC to develop large programs.

**STRUCTURED** Create well organized programs using procedures and functions that are easily identified and understood and completely reusable in future programs.

**MODULAR** Use procedures and functions grouped together to form "library modules."

**INTERACTIVE** BetterBASIC acts like an interpreter, responding to the users' commands in an immediate mode. However, each statement is actually compiled as it is entered.

**EXTENSIBLE** Create your own BetterBASIC modules which contain BetterBASIC extensions (ideal for OEMs).

**COMPILED** Each line of the program is compiled as it is entered into the computer's memory rather than interpreted at runtime. The optional Runtime System generates EXE. files.

**BetterBASIC** Runs on IBM PC, IBM PC/XT and compatibles.

**CALL 1-800-225-5800** Order Better BASIC now, or write Summit Software Technology, Inc.™, P.O. Box 99, Babson Park, Wellesley, MA 02157. Prices are listed below.

BetterBASIC: $199 Runtime System: $250
8087 Math Module: $99

Still not convinced? Order the BetterBASIC sample disk which includes a demo, a tutorial, compatibility issues, 50 lines of BetterBASIC and more. Only $10.

MasterCard, VISA, P.O. Checks, Money Order, C.O.D. accepted.

BetterBASIC is a registered trademark of Summit Software Technology, Inc.

IBM PC and IBM PC/XT are registered trademarks of International Business Machines Corp. Tandy is a registered trademark of Tandy Corp. Illustrated above are registered trademarks of the following companies: Mobil Oil Corp.; A T & T; General Electric Co.; Westinghouse Electric Corp.; TRW, Inc.

Better BASIC™

ALSO AVAILABLE FOR THE TANDY 2000, 1200 AND 1000

CIRCLE 71 ON READER SERVICE CARD

# Slash Programming Time in Half!

## With FirsTime™

- Fast program entry through single keystroke statement generators.
- Fast editing through syntax oriented cursor movements.
- Dramatically reduced debugging time through immediate syntax checking.
- Fast development through unique programmer oriented features.
- Automatic program formatter.

### FirsTime is a true syntax directed editor.

FirsTime ensures the integrity of your programs by performing all editing tasks like moves, inserts and deletes along the syntactic elements of a program. For example, when you move an IF statement, FirsTime will move the corresponding THEN and ELSE clauses with it.

Even FirsTime's cursor movements are by syntax elements instead of characters. The cursor automatically skips over blank spaces and required keywords and goes directly to the next editable position.

### FirsTime is a Syntax Checker

FirsTime checks the syntax of your program statements, and also:

- Semantics like undefined variables and mismatched statement types.
- The contents of include files and macro expansions.
- Statements for errors as they are entered and warns you immediately.

### FirsTime is a Program Formatter

FirsTime automatically indents statements as they are entered, saving you from having to track indentation levels and count spaces.

### FirsTime has Unique Features

No other editor offer these features:

The *Zoom command* gives you a top down view of your program logic.

The *View command* displays the contents of include files and macro expansions. This is valuable to sophisticated programmers writing complex code or to those updating unfamiliar programs.

FirsTime's *Transform command* lets you change a statement to another similar one with just two keystrokes. For example, you can instantly transform a FOR statement into a WHILE statement.

The *Move at Same Level command* moves the cursor up or down to the next statement at the same indentation level. This is very useful. For example, you can use it to locate the ELSE clause that corresponds to a given THEN clause or to traverse a program one procedure at a time.

### FirsTime is Unparalleled

FirsTime is the most advanced syntax directed editor available. It makes programming faster, easier and more fun.

**TO ORDER CALL (201) 741-8188**

or write:

## *Spruce Technology Corporation*

189 E. Bergen Place
Red Bank, NJ 07701

In Germany, Austria and Switzerland contact:

Markt & Technik Software Verlag
Munchen, W. Germany
(089) 4613-0

FirsTime is a trademark of Spruce Technology Corporation • MS-DOS is a trademark of Microsoft Corporation
IBM is a trademark of International Business Machines, Inc. • Turbo Pascal is a trademark of Borland International

CIRCLE 26 ON READER SERVICE CARD

FirsTime for Turbo Pascal $74.95

FirsTime for MicroSoft & IBM Pascal $245

FirsTime for C $295

# SuperSoft Languages When Performance Counts

A programmer's most important software tool is the language compiler or interpreter he uses. He has to depend on it to work and work well.

At SuperSoft, we believe it. That's why we offer four excellent compilers: SuperSoft FORTRAN, SuperSoft A, SuperSoft C, and SuperSoft BASIC. They answer the programmer's need for rock solid, dependable performance on microcomputers.

## SuperSoft FORTRAN

**With large code and data.**
SuperSoft FORTRAN version 2.0 with large code and data space is now available under MS DOS and PC DOS. It gives you the power to compile extremely large FORTRAN programs on micros. It allows double precision and complex numbers, full IEEE floating point, and a full range of other important features for the serious FORTRAN programmer. Both 8087 support and a RATFOR preprocessor are optionally available.

FORTRAN (CP/M-80 & 86, MS DOS, PC DOS): $325
8087 support: $50 RATFOR: $100

## SuperSoft A

**A true Ada* subset**
SuperSoft A is a completely standard subset of the Ada language, incorporating approximately 63% of the standard Ada syntax and including such important features as packages and separate compilation. For CP/M-80 microcomputers: $300.

## SuperSoft C

SuperSoft C is a high-powered, full-featured C compiler designed for serious C applications. It is fast – both in compilation and execution, and it is packed with more than 200 library functions (all delivered in source code form). SuperSoft C produces optimized assembly code, and object code can be ROMed.

SuperSoft C (for CP/M-80, CP/M-86, MS DOS, PC DOS): $350

Languages
BASIC Compiler
Languages
Fortran
Languages
C Compiler
SuperSoft
SuperSoft

Languages
BASIC Compiler
SuperSoft
SuperSoft

## SuperSoft BASIC

The SuperSoft BASIC compiler lets you get serious with business and financial programs. It uses BCD math to give you highly accurate results for demanding applications. SuperSoft BASIC is a true native code compiler that is generally compatible with Microsoft's BASIC interpreter. And an additional bonus – no run time license fee is required.

SuperSoft BASIC Compiler (for MS DOS, PC DOS, and CP/M-86): $300

**Also available for programmers:**

Star-Edit, a full-featured programmer's text editor: $225.00
Disk-Edit, an invaluable programmer's disk data editor: $100.00

To order call: **800-762-6629**

In Illinois call 217-359-2112

**SuperSoft**

**SuperSoft, Inc.,** 1713 S. Neil St.,
P.O. Box 1628, Champaign, IL 61820

In conjunction with SuperSoft, SuperSoft FORTRAN was developed by Small Systems Services, Urbana, IL, a leader in FORTRAN development.

**Japanese Distributor:** ASR Corporation International, TBL Building, 7th Floor, 1-19-9 Toranomon, Minato-Ku, Tokyo 105, Japan Tel. 03-5025550. Telex 222-5650 ASRTYO J.

*Ada is a trademark of the Department of Defense
PC DOS is a trademark of International Business Machines.
MS DOS is a trademark of Microsoft.
CP/M-80 and CP/M-86 are trademarks of Digital Research, Inc.

CIRCLE 74 ON READER SERVICE CARD

# Dynamic hashing in external searching

**By Namir Clement Shammas**

We resume last month's discussion on hashing by presenting dynamic hashing techniques used in external searching.

All the techniques presented last month had a common disadvantage—the necessity of predetermined hash table sizes. If the table is undersized, access speed will increase. In contrast, an oversized hash table will yield an unacceptably low storage utilization. What methods can be used to allow the hash table to increase in size as needed, and what are the criteria for this?

First, let us briefly go over the data structures involved. We are storing search keys in file areas called buckets. The number of keys per bucket depends on their relative sizes.

There are two types of buckets: primary buckets, whose addresses are calculated by the hashing function, and overflow buckets. As primary buckets fill up, additional keys are stored in the overflow buckets, forming a chain of linked lists. Each of the linked lists contains one primary bucket and a number of overflow buckets. The more overflow buckets, the slower the search is. This method is called chaining.

Let us begin by focusing on what's necessary to increase the hash table size. It depends on the average length of search (*ALOS*) and a maximum allowable value, *ALOS_MAX*. The *ALOS* is calculated from:

$$\text{ALOS} = \text{Sum of all } (i+1)\ N(i)\ /\ T \text{ for } i = 0,1,2,\ldots,\text{max\_limit}$$

where *N(0)* is the number of keys in all of the primary buckets. *N(i)* (for $i > 0$) is the number of keys in the *i*th overflow bucket. And *T* is the sum of all *N(i)* for *i* = 0,1,2, . . . ,*max_limit*.

Now we turn our attention to the methods used for dynamic hashing. They are:

**Linear hashing.** This method doubles the size of the hash table when the *ALOS* value exceeds *ALOS_MAX*. If the initial table size is *S0* buckets, then the size at any time, *S*, is $S = 2^d S0$, where *d* is the doubling factor. With the table size increasing, we are also creating a series of hashing functions with an increasing address range.

Listing 1 shows Programmer's Pseudo Listing (PPL) code for a procedure to add new keys and possibly increase the table size. As the listing suggests, the *ALOS* value is monitored as new keys are added.

When the table size is increased for the first time, the first primary bucket, *BUCKET(0)*, is split. This is done by taking the second half of the keys and moving them *S0* addresses into the newly created bucket number *S0*. The *ALOS* value is recalculated. If it is still more than *ALOS_MAX*, the second primary bucket is split and so on until *ALOS* becomes less than *ALOS_MAX*.

Buckets are split if more than one key is in them, otherwise they are skipped. The sequential splitting will stop before all of the primary buckets are split. The address of the next primary bucket to be split is preserved. It will be used the next time bucket splitting is carried out. Once the last primary bucket is reached, the pointer is reset to point to the first bucket. Generally, keys are moved $2^d$ *S0* addresses.

Once splitting occurs, this method will cause the primary buckets to exist in three types: split, unsplit, and new buckets. Under these conditions, the two most recent hashing functions are needed to locate keys.

Listing 2 shows PPL code for searching in a linear hashing scheme. As shown, we use the previous hashing function to calculate the bucket address, then check whether the bucket was unsplit. If it is not split, we recalculate the address using the most recent hashing function.

**Linear hashing with partial expansions.** This method is an improvement over the first one. The scheme used in doubling the hash table size maintains a more uniformly distributed hash table. We will consider doubling using two partial expansions.

Listing 3 shows PPL code for this method. Each expansion stage adds 50% of the table size and is composed of a number of passes.

In the first stage, each pass will move selected keys from two buckets (Figure 1) to a newly created one. This continues in a wavelike fashion until half of the original table size is added.

The second stage is similar to the first, except that in each pass, selected keys from three buckets are moved to a new bucket. This is carried out until the hash table size doubles.

**Virtual hashing.** This method is similar to linear hashing in that it doubles the hash table size. Its differences with linear hashing are as follows:

- Each primary bucket has an in-use flag or bit.
- The scheme for doubling the table size and splitting primary buckets is different.
- The entire sequence of hashing functions generated may be used to locate a bucket, starting with the most recent one and going back to previous ones.

Listing 4 shows PPL code for adding a new key to a system using virtual hashing. First, the hashing function with a current doubling factor is used to give a bucket address. The corresponding flag indicates whether it is occupied. If it is vacant, the previous hashing function is used to obtain another address. The operation continues until an occupied bucket is located.

**Dynamic hashing.** This method uses three data structure components to accommodate the increase in keys. The first component is a hash table with a predefined and fixed size. The second is the buckets containing search keys. No overflow buckets are needed. The third is a dynamically growing set of trees with the task of recording the split in buckets.

The root of each tree is a hash table address. Thus, there can be as many trees as the table size. Initially, an equal number of buckets and hash table addresses exist, while the dynamic split trees are empty or nonexistant.

As keys are added, the hashing functions provide the address of the hash table which in turn points to the appropriate bucket. When buckets start to fill, the hash table addresses become the roots of new split trees.

The two new tree leaves are numbered zero and one. Leaf number zero points out to the recently filled bucket, while leaf number one points to the newly created bucket. This scheme of creating and numbering new leaves is used to build the split trees.

Questions arise concerning how the keys are distributed during a split and how to maintain a correct search algorithm. In other words, how do we travel through a split tree and locate the correct bucket?

One method is to use a second hashing function for the split trees. The function is based on a pseudo random number generator yielding zeros or ones. The search key is used to provide the seed for this function. This will provide a consistent sequence of zeros and ones needed to traverse the split trees.

Listing 5 shows PPL code for adding a new key to this hashing method. The same listing contains a procedure *Search_Bucket* that is used to traverse through the split trees.

**Modified dynamic hashing using deferred splitting.** This technique is a modification over the previous method. A drawback of dynamic hashing is that the storage location for the split trees will quickly become large.

A way to solve this problem is to use an area for overflow buckets linked to the primary buckets. Each primary bucket has a counter for the number of overflow buckets linked with it. As the number exceeds a preset value, splitting occurs, following the same steps as with the previous method. This method offers more efficient storage utilization.

Hashing techniques are very versatile methods for fast searching. The subject is very interesting and vast. It is worthy of additional attention and further discussions. If you have any questions, comments, or would like to share other methods and code, please write to me in care of *COMPUTER LANGUAGE*, or drop me a line on the *COMPUTER LANGUAGE* Bulletin Board Service or CompuServe.

You may wonder what other search techniques rival the ones we have discussed. The answer will be presented in the next issue. We will discuss B-trees and the improved version, the B+ tree. We will also discuss a new improvement on the B+ tree.

**References**

1. Tremblay, J.P. and Sorenson, P.G. *An Introduction to Data Structures with Applications*. Second Edition, McGraw-Hill Book Co., New York, N.Y., 1984.
2. Kruse, R.L. *Data Structures and Program Design*, Prentice-Hall, Englewood Cliffs, N.J., 1984.

**Linear hashing with partial expansions**

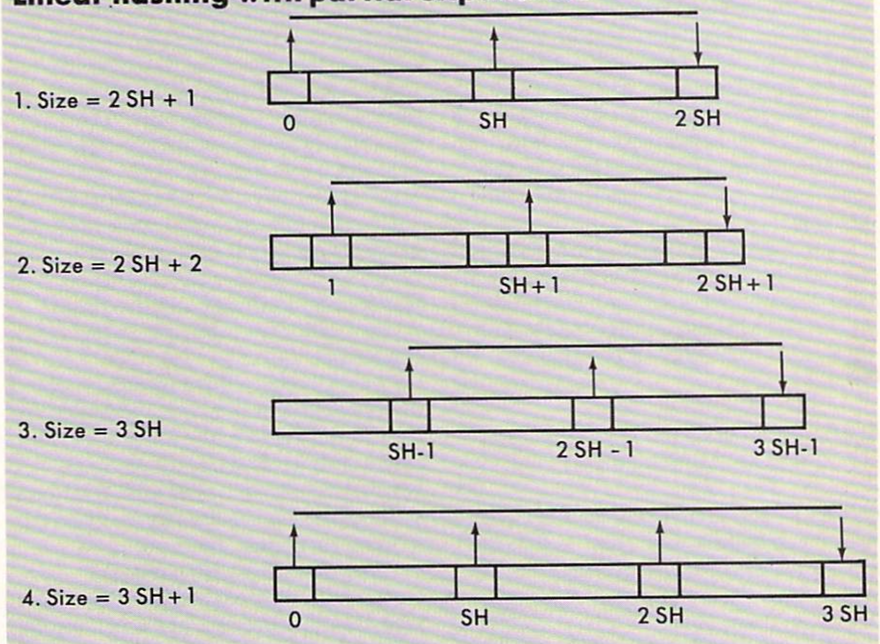


Figure 1.

## PPL updates

It seems the subject and concept of PPL struck a chord with many readers. I received a lot of feedback suggesting additional and modified syntax.

One reader stressed the virtues of indenting PPL code and asked that it become mandatory. Another reader recommended PPL code start with a description of the program's task, references, version number and any other relevant information. This is desirable especially in big projects.

A few readers suggested dropping the square and curly brackets that surround the names of procedures and functions, *if* clarity is maintained. This can be done especially when PPL presents rather detailed code and uses formal comments for narratives. I still recommend using these brackets when PPL code presents a combination of English written code and includes highlighted function or procedure calls. If using the program NILE to perform syntax checking, then using the brackets is still mandatory.

Concerning function recursion, a reader suggested the following: if a function explicitly calls itself, then use the declaration:

```
EXPLICITLY RECURSIVE FUNCTION function_name
```

On the other hand, if the function calls another function, which in turn calls the first one, then use the following:

```
IMPLICITLY RECURSIVE FUNCTION function_name
```

These declarations add more clarity and show how recursion is used and the way it occurs.

William Blum, Daly City, Calif., sent me the PPL listing of a screen editor (PANEL.PPL), another for structural program analysis (STRANA.PPL), a C listing for a PPL pretty printer (PPLI.C), and an enhanced version of the Pascal program NILE to include an automatic time stamp for the PPL code for systems running under MS-DOS. These listings will be available on the *COMPUTER LANGUAGE* BBS and CLM-SIG in CompuServe. Many thanks to Bill, who shared these listings with us.

```
PPL code for linear hashing

PROCEDURE Add_Key1(New_Key : Key_Data)

-- All identifiers are assumed global
-- ALOS is the Average Length Of Search, and ALOS_MAX is its maximum value.
-- BUCKET() is the array of buckets
-- B is the number of keys in a primary area bucket
-- S0 is the initial size of the hash table.  The actual size is calculated
--     using the following S = S0 * 2^(Doubling_factor)
-- Doubling_factor is used to expand the hash table size, and is initially
--     set equal to zero.
-- Next_Bucket is used to indicate the next bucket to be split as ALOS
--     exceeds its maximum allowable value. Initially equal zero.

BEGIN
    INITIALIZE: Calculate ALOS
                Update_Flag = FALSE
    LOOP
    BEGIN  IF ALOS > ALOS_MAX THEN   EXIT  END IF
        IF keys in BUCKET(Next_Bucket) > 1
        THEN
            Create BUCKET(S0 + Next_Bucket)
            Move last (B/2) keys from BUCKET(Next_Bucket)
                                 to BUCKET(S0 + Next_Bucket)
            Calculate new ALOS value
        END IF
        Next_Bucket += 1
        Update_Flag = TRUE
    END LOOP
    TERMINATE: ADDRESS = Hash_function(New_Key,Doubling_factor,S0)
               -- Update factor for new range of addresses
               IF Update_Flag THEN Doubling_factor +=1 END IF
END Add_Key1
```

Listing 1.

```
PPL code for searching in a linear hashing scheme

FUNCTION Search_Key1(Sought_Key : Key_Data;    S0 : INTEGER;

BEGIN
    IF Doubling_factor = 0
    THEN
        ADDRESS = Hash_function(Sought_Key,Doubling_factor,S0)

    ELSE -- Assume key is located in a bucket that was not split
      ADDRESS = Hash_function(Sought_Key,(Doubling_factor - 1),S0)
      IF (ADDRESS < Next_bucket) AND (Sought_Key is not in BUCKET(ADDRESS))
      THEN
          ADDRESS = Hash_function(Sought_Key,Doubling_factor,S0)
      END IF
    END IF
    RETURN (ADDRESS)
END Search_Key1
```

Listing 2.

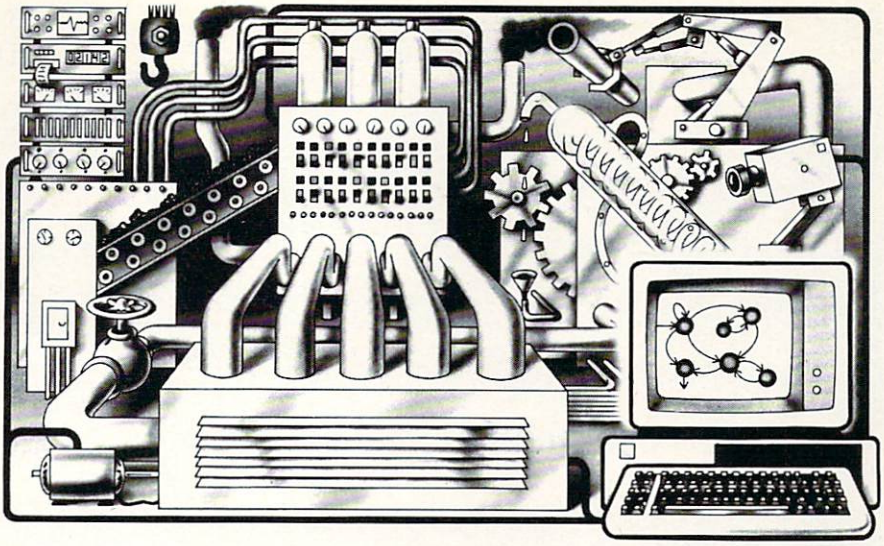
```
PPL code for linear hashing with two partial expansions

PROCEDURE Expand;
-- Two-stage partial expansion scheme
-- SH =  (number of initial buckets) / 2

BEGIN
    INITIALIZE: SH = SO / 2
    LOOP
    BEGIN   FOR I = 1 TO SH
        N = 2 * SH + I
        Create BUCKET(N)
        Move selected keys from BUCKET(I - 1) to BUCKET(N)
        Move selected keys from BUCKET(SH + I - 1) to BUCKET(N)
    END LOOP
    TERMINATE: None

    INITIALIZE: None
    LOOP
    BEGIN   FOR I = 1 TO SH
        N = 3 * SH + I
        Create BUCKET(N)
        Move selected keys from BUCKET(I - 1) to BUCKET(N)
        Move selected keys from BUCKET(SH + I - 1) to BUCKET(N)
        Move selected keys from BUCKET(2 * SH + I - 1) to BUCKET(N)
    END LOOP
    TERMINATE: SO *= 2     -- double table size.
END Expand
```

Listing 3.

```
PPL code for virtual hashing

PROCEDURE Add_Key2(New_Key : Key_Data)

-- Identifiers used are similar to those in Add_Key1.
-- Used_Flag() is an array of bits or flags to indicate if a bucket is
--             occupied by at least one key.

BEGIN
    INITIALIZE: ADDRESS = Search_Key2(Sought_Key,Doubling_factor,SO,N)
    LOOP
    BEGIN IF BUCKET(ADDRESS) is not full THEN EXIT END IF
        IF N = Doubling_factor
        THEN
            IF ALOS > ALOS_MAX THEN Doubling_factor += 1
                               ELSE -- no need to double size, yet
                                    Obtain overflow address
            END IF
        END IF
        Used_Flag(ADDRESS + 2^N * SO) = TRUE
        N += 1
        Rehash keys in BUCKET(ADDRESS) and linked overflow buckets, using N
        ADDRESS = Hash_function(Sought_Key,N,SO)
    END LOOP
    TERMINATE: Add New_Key to BUCKET(ADDRESS)
END Add_Key2
```

Listing 4 *(Continued on a following page).*

# Csharp Realtime Toolkit



**Realtime on MSDOS? Csharp** can do it! Get the tools without operating system overhead. Cut development time with C source code for realtime data acquisition and control. **Csharp** includes: graphics, event handling, procedure scheduling, state system control, and interrupt handling. Processor, device, and operating system independent. **Csharp** runs standalone or with: MSDOS, PCDOS, or RT11. **Csharp** runs on: PDP-11 and IBM PC. **Csharp** includes drivers for Hercules and IBM graphics boards, Data Translation and Metrabyte IO boards, real time clock, and more. Inquire for Victor 9000, Unix, and other systems. Price: $600

**SYSTEMS GUILD**

Systems Guild, Inc., P.O. Box 1085, Cambridge, MA 02142
(617) 451-8479

**CIRCLE 27 ON READER SERVICE CARD**

---

***Another in a series of productivity notes on UNIX™ software from UniPress.***

*CROSS COMPILER FOR THE 8086™ FAMILY*

## LATTICE® C CROSS COMPILER

***Subject: C Cross Compiler for the 8086 Family.***

*The Lattice C Cross Compiler allows the user to write code on a VAX™ (UNIX or VMS™) or MC68000™ machine for the 8086 family. Lattice C is a timesaving tool that allows a more powerful computer to produce object code for the IBM-PC™. The compiler is regarded as the finest C compiler for the 8086 family and produces the fastest and tightest code.*

***Features:***

- ■ *For your UNIX or VMS Computer.*
- ■ *Use your VAX or other UNIX machine to create standard Intel object code for the 8086 (IBM-PC).*
- ■ *Highly regarded compiler produces fastest and tightest code for the 8086 family.*
- ■ *Full C language and standard library, compatible with UNIX.*
- ■ *Small, medium, compact and large address models available.*
- ■ *Includes compiler, linker, librarian and disassembler.*
- ■ *8087™ floating point support.*
- ■ *MS-DOS™ 2.0 libraries.*
- ■ *Send and Receive communication package optionally available. Price $500.*
- ■ *Optional SSI Intel Style Tools. Package includes linker, locator and assembler and creates executables for debugging on the Intel workstation or for standalone environments. Price $8,550.*

***Price:***

| | |
|---|---|
| *VAX (UNIX or VMS)* | *$5000* |
| *MC68000* | *3000* |

*For more information on these and other UNIX software products, call or write: UniPress Software, Inc., 2025 Lincoln Hwy., Edison, NJ 08817. Telephone: (201) 985-8000. Order Desk: (800) 222-0550 (Outside NJ). Telex: 709418. Japanese Distributor: Softec 0480 (85) 6565. European Distributor: Modulator SA (031) 59 22 22.*

*OEM terms available.*
*Mastercard/Visa accepted.*

*Trademarks of: Lattice, Lattice, Inc.; VAX and VMS, Digital Equipment Corp.; UNIX, AT&T Bell Laboratories; IBM PC, International Business Machines; MS DOS, Microsoft; MC68000, Motorola; 8086/8087, Intel*

**UniPress** Software
*Your Leading Source for UNIX Software.*

**CIRCLE 81 ON READER SERVICE CARD**

```
PPL code for searching in a virtual hasing scheme

FUNCTION Search_Key2(Sought_Key : Key_Data; Doubling_factor, SO : INTEGER;
                                           VAR N : INTEGER) return INTEGER

BEGIN
    N = Doubling_factor
    INITIALIZE: ADDRESS = Hash_function(Sought_Key,N,SO)
    LOOP
    BEGIN IF Used_Flag(N) THEN EXIT END IF
        N -= 1;
        ADDRESS = Hash_function(Sought_Key,N,SO)
    END LOOP
    TERMINATE: RETURN(ADDRESS)
END Search_Key2
```

**Listing 4** *(Continued from a preceding page).*

```
Adding keys to a dynamic hashing scheme

PROCEDURE Add_Key3(New_Key : Key_Data)

-- Add a new key using dynamic hashing scheme

BEGIN
    ADDRESS = Hash_function(New_Key,Hash_Table_Size)
    IF Tree_ptr(ADDRESS) is nul
    THEN -- Hash table address points to bucket directly
        Process_Bucket
    ELSE -- Hash table address is a root for a split tree
        Search_Bucket -- obtain bucket address
        Process_Bucket
    END IF
END Add_Key3

----------------------------------------------------------------------

PROCEDURE Search_Bucket

-- Search for a bucket by traversing a split tree

BEGIN
    INITIALIZE: ROOT = Hash_Ptr(ADDRESS)
    LOOP <Traverse>
    BEGIN
        ADR = Hash_function2(New_Key)
        IF Leaf(ROOT,ADR).Type is terminal THEN EXIT <Traverse> END IF
        ROOT = Leaf(ROOT,ADR).Ptr
    END LOOP <Traverse>
    TERMINATE: ADDRESS = Leaf(ROOT,ADR).Ptr
END Search_Bucket
```

**Listing 5** *(Continued on a following page).*

# THE CLASSICS AND THE STATE-OF-THE-ART

## NEW WAYS TO KEEP UP WITH PROGRAMMING LANGUAGES

### *Programming in Modula-2*

***3rd Edition***

**Niklaus Wirth**

Reviews of the second edition:
"It is remarkable that so powerful a language can be described so concisely.... As the language becomes more important, [this book] will serve as the arbiter of what constitutes correct usage...."
— D. D. Clark, BYTE, 8/84

"This has the clarity of a master speaking on his favorite topic.... The discussions are full of insight into the considerations that go into building a correct program...."
— Computing Reviews

1985/approx. 200 pp./Cloth $18.00 (tent)
(Texts and Monographs in Computer Science)
ISBN 0-387-15078-1

### *Programming in Prolog*

***2nd Edition***

**W. F. Clocksin** and **C. S. Mellish**

Review of the first edition:
"An excellent guide to the language."
— Compute!

Corrected and up-dated, this new edition uses examples to demonstrate how useful programs can be written with the Prolog system that exists today.

1985/297 pp./Paper $17.95
ISBN 0-387-15011-0

### *Pascal: User Manual and Report*

***Third Edition***

**K. Jensen** and **N. Wirth**
Revised by **A. Mickel** and **J. Miner**

The original description of the language has been up-dated to incorporate the ISO-standard. Also included in this reworked edition is Wirth's EBNF syntactical notation; an expanded user manual with improved programming examples; and a detailed appendix summarizing changes in the language necessitated by the ISO-standard.

1985/265 pp./76 illus./Paper $14.00
ISBN 0-387-96048-1

### *Modula-2 for Pascal Programmers*

**Richard Gleaves**

"A treatment of this type will let the working programmer 'come up to speed' quickly in this new language.... One of the best things about **Modula-2 for Pascal Programmers** is that each important language feature is illustrated with at least one example of correct usage... Aside from the profuse examples, Mr. Gleaves provides notes and warnings at many locations, pointing out not-so-obvious features of the operation of the language...."
— D. D. Clark, BYTE, 11/84

1984/145 pp./18 illus./Paper $16.95
(Springer Books on Professional Computing)
ISBN 0-387-96051-1

### *The American Pascal Standard*

***With Annotations***

Edited by **Henry Ledgard**

This complete ANSI/IEEE Pascal standard is supplemented by Henry Ledgard's annotations, which makes the new American Pascal standard accessible to the programmer and enduser alike.

1984/97 pp./Paper $16.95
(Springer Books on Professional Computing)
ISBN 0-387-91248-7

### *Forth*

**W. Salman**, **O. Tisserand**, and **B. Toulout**

Now you can get the benefit of efficient, economical programs from using Forth. This systematic volume takes you through the history, describes the fundamentals, and explains important applications. Each step is supplemented by solved problems.

1985/approx. 225 pp./Paper $14.00
ISBN 0-387-91256-8

**Springer-Verlag**
New York Berlin Heidelberg Tokyo

### *Problem Solving Using UCSD Pascal*

***2nd Edition***

**K. L. Bowles**, **S. Franklin**, and **D. Volper**

1984/340 pp./106 illus./Paper $17.95
ISBN 0-387-90822-6

### *Ada: An Introduction*

***2nd Edition***

**Henry Ledgard**

1983/135 pp./1 illus./Paper $14.95
ISBN 0-387-90814-5

### *A Practical Introduction to Pascal*

***2nd Edition***

**I. R. Wilson** and **A. M. Addyman**,

1982/236 pp./46 illus./Paper $14.95
ISBN 0-387-91210-X

### *Send For Your Free Computer Language Package Today*

___Please rush me my free information package describing some of the best computer language titles.

Name ____________________

Address ____________________

City/State/Zip ____________________

I am particularly interested in:

____________________

____________________

Springer-Verlag New York, Inc. S938
175 Fifth Avenue
New York, NY 10010
Attn: David Dwek

CIRCLE 70 ON READER SERVICE CARD

```
PROCEDURE Process_Bucket

BEGIN
    READ Bucket_File,ADDRESS,BUCKET
    IF BUCKET.Count_Key = MAX_KEY
    THEN -- BUCKET(ADDRESS) is full
        Split_Bucket
    ELSE -- Simply add key
        BUCKET.Count_Key += 1
        BUCKET.Key(BUCKET.Count_Key) = New_Key
    END IF

END Process_Bucket

PROCEDURE Split_Bucket

BEGIN
    Num_Bucket += 1
    BUCKET.Key(MAX_KEY+1) = New_Key
    INITIALIZE: BUCKET.Count_Key = 0
                NEWBUCKET.Count_Key = 0
    LOOP
    BEGIN FOR i = 1 to MAX_KEY + 1
        ADR = Hash_Function2(BUCKET.Key(i)) -- 0/1 function
        IF ADR = 0
        THEN BUCKET.Count_Key += 1
            BUCKET.Key(BUCKET.Count_Key) = BUCKET.Key(i)
        ELSE
            NEWBUCKET.Count_Key += 1
            NEWBUCKET.Key(BUCKET.Count_Key) = BUCKET.Key(i)
        END IF
    END LOOP
    TERMINATE: WRITE Bucket_File,ADDRESS,BUCKET
               WRITE Bucket_File,Num_Bucket,NEWBUCKET
               Create leaves in memory and link them hash table
               Set leaf pointers to ADDRESS and Num_Bucket
END Split_Bucket
```

Listing 5 *(Continued from a preceding page).*

# PUBLIC DOMAIN SOFTWARE REVIEW

## Macintosh and MacBASIC

### By Tim Parker

Few machines have been the subject of as much controversy as the Macintosh. People seem to either love it or hate it.

Certainly, few machines can match its graphic approach to computer use with as much user friendliness. Unfortunately, the machine pays its price in execution speed. The Mac may have been the machine for the rest of us, but anyone wanting to do anything really productive with the Mac (such as business applications or word processing) had better be willing to sacrifice time.

Experienced computer users can look at the Mac in two ways. First, it is undoubtedly a fun machine. The graphical layout of everything concerned with the Mac makes it interesting to play with.

The appearance of icons and mouses (mice?) makes a welcome change from the text-oriented computers we are used to. However, the low-level approach tends to wear thin on the patience after a while. The sheer process of having to swap disk after disk and wait for an application to load often results in veterans relegating the Mac to a curiosity.

For novice computer users, though, the Mac allows a better introduction to the computer environment than any other machine. When the time comes to upgrade to more power, there may be a few problems with adapting to more normal operating systems, but the experience of using the Mac should help make the transition smoother.

The Mac has one outstanding feature that cannot be faulted—its excellent graphics. The majority of applications for the Mac tend to draw on this feature, and programming for the Mac can be a lot of fun. It is not surprising, therefore, that most home programmers have written graphic-oriented programs.

The public domain material available for the Mac has been interesting to follow. It started off as a very loose collection of odds and ends, as most machines inspire. But the Mac was excluded from mass distribution of public domain material by not having any group pull together available material and offer it nationally.

Of course, user groups sprang up everywhere with local libraries of programs. Networks such as CompuServe and the Source became populated with Mac programs also, but the majority of Mac owners have access to neither. It brought some relief when one group assembled some of this material together and offered it at a reasonable price to a national audience.

The Public Domain Exchange has been mentioned here before as a source for Apple programs. It seemed a logical step then to cover the Mac, and they have done so.

The Public Domain Exchange took Mac material from the networks (notably CompuServe) and several larger user groups and had released 24 disks of Mac programs when I wrote this article. The disks are available for a paltry $10 each, which, considering the cost of the tiny disks, is quite an excellent bargain. Service is rapid, and the staff is willing to talk on the phone.

Since the Mac is still in its early growth, there isn't yet an equivalent to the excellent programs available for the IBM PC or CP/M machines where, for example, word processors, spreadsheets, communications packages, and languages that rival commercial products are available. But this is simply a matter of time.

So what has the Public Domain Exchange to offer Mac users? Quite a lot, in fact, most of which will be covered in this and an upcoming column. However, since this is a BASIC issue, we may as well begin with BASIC programs for the Mac.

The Mac relies on a version of Micro-Soft BASIC that accommodates the graphic capabilities of the machine. It is relatively small (47K) and allows a reasonable amount of programming power. Some idiosyncrasies exist, but few languages escape that.

Mac disk no. 1 from the Public Domain Exchange is full of MS-BASIC programs mostly off CompuServe. They are loosely grouped into three categories: demonstrations, utilities, and business and home use. A Read.Me file has an introduction to the disk while a Program Notes file has documentation of the disk's contents. It does not require MacWrite to read this and other files on the disk, as an editor is included. The documentation is rather limited, consisting of the summary file from CompuServe in most cases. These, for those who haven't been on CompuServe, are usually three or four lines long and simply describe the program's purpose.

When the disk is started, a nice graphics picture appears while the disk is being attacked by Finder, the operating system. The six utility programs are of varying use. Benchmark runs a 10-line BASIC program that loops through a sequence and evaluates exponentials and square roots. Quite honestly, I don't know why this program was included. I had hoped for a more substantial benchmark such as a Sieve of Eratosthenes.

Macwidth will reformat an ASCII text file into any width desired. MacTEP++ is an extension of the Mac terminal emulator program that adds extra features such as autodial/redial, baud rate reset, timer, and others. Dskdmp is a disk dump utility that allows dumping and writing from a buffer or address dumping. It may be useful when disk errors are encountered or when a queue is required for some file saving tricks.

Edit is a very limited text editor written in BASIC. The rationale behind another text editor for the Mac, especially in BASIC, seems curious. It must be assumed it was a programming exercise. MacCopy is a useful disk copy routine that shows up on several different user group disks. Version 1.3 is included here.

The demonstration programs comprise five routines, none of which will amaze anyone. Animation, when started, produced a ball on the left side of the screen that bounced to the right side and disappeared, never to re-emerge. I figured something was wrong so I ran the program again, producing exactly the same result. *LIST*ing the code proved that the program did what it was designed to do.

Globe draws a spherical model about 2 in. (5 cm) in diameter with great circle lines marked. This then rotates. It is a nice demonstration of smooth graphics in BASIC, if somewhat limited.

Mouse, which may be assumed to have something to do with the hardware device of the same name, produces a menu that resembles a communications program front end. All attempts to get it to work failed. Even trying to exit the system didn't work. All that happened was that a secondary menu reappeared.

Clock produces an analog clock face on the screen with a sweep second hand. It kept reasonable time but was surpassed by a program to be mentioned shortly. Flash didn't seem to do anything except erase the screen until the source code revealed it was designed to "negate the screen quickly."

The business and home programs were a little better. They mostly comprised home finance material, such as home budgeting, interest calculations, and lease/buy decisions. The rest were statistics packages for several routines, including unbiased standard deviations, chi-square, and others.

Finally, the disk contains a Screen Maker that allows a user to save and recall screen images to disk. All in all though, the disk was not quite the package one could hope for. The terminal extension was by far the most interesting and useful item on the disk and probably worth it by itself. The business programs were fairly good too, but treat the demonstration programs as filler.

Another MS-BASIC disk is Mac disk no. 12. This one was a lot of fun and occupied a good afternoon of examination. A few of the programs on the disk are used as standard demonstration material at a local Mac dealer.

The disk is divided into a number of folders. The Bits and Pieces subdirectory has two items in it, a program called Desk Cal and a MacASM Demo.

The MS-BASIC folder was a real treat. It was further divided into Games and Demo, Programming Aids, and MS-BASIC Sound. Although games are usually treated rather lightly in this column (real programmers would rather write them then play them, right?), the capabilities of the Mac are ideally suited to games, so, in the interest of authoritative journalism, I ventured into the Games area.

Go is a good version of the ancient game. Documentation for this and several other programs on the disk are in a folder for MacWrite. Clock is the same program mentioned earlier. Abe & Mona really impressed me. It allows high-resolution pictures of Abraham Lincoln and the Mona Lisa to be drawn in block graphic chunks. The graphics were superb.

This program is one of the most popular demos to show off the Mac's graphic capabilities, followed closely by Living Art, which shows a line graphic continually moving around the screen. It reminded me of the video game QIX, although it wasn't a game. The results of the overlapping pixel designs were interesting.

Finally, ABM-ICBM is based on the video game Missile Command. It draws the screen in the upper left third (somewhat limited but tolerable), and the mouse is used to move a pointer around. Missiles are fired with three keys from the keyboard (a, s, and d) at incoming bombs intent on destroying your cities.

The Sound folder contains four music programs that use the Mac as a four-tone synthesizer. The coding is very interesting to examine, as the waveforms, music tables, and more are assembled. The music available on disk is Anton Dvorak's New World Symphony (No. 9), a Bach minuet from Anna Magdelena's notebook, and two bits from Star Wars: the main theme and Princess Leia's theme. Turn the sound control up.

The Programming Aids folder was stocked with a wide variety of items, including a Calculator, a program to strip carriage returns from text files, a file comparison program, a program to strip REM statements, a Dvorak keyboard redefinition, a pattern editor, and a font displayer.

A disassembler rounds it all out. This disk was the antithesis to the first and is very highly recommended to all. Wait 'till you see Mona.

Mac disk no. 3 contains a few programs worth a third or fourth glance. Clock displays the time in a 24-hour format as digital numbers on the screen. The transition from one number to another is via smooth animation. The program was fairly accurate at time keeping. The entire screen is blank except for the letters in white on gray. The only fault I found was the lack of an exit sequence, unless it is hidden.

Amazing is a cute game that draws a maze on the screen. The user can select four difficulty levels. Level one is really simple, while the highest was almost impossible. Luckily, a pull-down menu allows the answer to be displayed, then erased. The maze, once drawn, is navigated with the mouse, with the path highlighted. This program was rather fun and can provide a lot of amusement for some of the younger members of the audience.

INTERACTIVE
PASCAL
Introductory Offer
Now Only $39.95

Announcing MYSTIC PASCAL—the interactive Pascal system. Command statements are INSTANTLY compiled and executed. As easy to use as Basic—AND its a true object code compiler!

- MYSTIC PASCAL is interactive—you can instantly compile and execute statements
- True 8088/8086 object code compiler
- We won't say how fast the compiler is, because you absolutely wouldn't believe it!
- Full screen editor for Standard Pascal programs
- Pop-Up Help windows describe the Standard Pascal language
- True MULTI-TASKING support
- Satisfaction guaranteed! 15 Day refund with no questions asked!
- Requires an IBM PC or true compatible, 256K, DOS 2.X, monochrome card

**MYSTIC CANYON SOFTWARE**
**P.O. Box 1010**
**Pecos, New Mexico 87552**

Place your order today!
Phone or use the coupon!
(505) 988-4214
10 to 5 Mountain Time

Name ____________
Address ____________
City ____________
State ________ Zip ________

Price is $39.95 plus $4 shipping. Outside US & Canada add $20 shipping. Payment must be in US funds on a US bank. Purchase orders accepted from recognized institutions.
NM residents add sales tax.

☐ Check/MO ☐ COD ☐ VISA ☐ MC

Card ____________
Exp. ____________

CIRCLE 57 ON READER SERVICE CARD

Two new fonts are included on this disk, Greek and Math. The Math font set was sorely needed for those who do any scientific work at all. Those in physics or chemistry will appreciate the Greek letters.

Soundlab allows music to be written on the screen and played with a four-channel synthesizer. It operates very similarly to the Music Construction Set from Electronic Arts. A piano keyboard occupies the center of the screen with various toggles above it and a list of notes. The mouse grabs a note duration and positions it on a set of staves. The playback is rather tinny but tolerable.

An interesting program is called Hendrix. This begins with a blank screen. Pressing the mouse's button and dragging the arrow across the screen results in electronic noises coming from the computer. As the mouse moves to the right, the pitch increases. Toward the left, it decreases.

Strange sounds can be produced by alternatively pressing the button and releasing it. The program has a built-in delay factor. The electronic screech is best heard with the volume turned up. Tricks such as rotating the mouse in a circle result in almost eerie sound effects. Sound duration is dependent on the mouse's sensitivity and the length of time the button is held down.

Rounding out the disk are a windowing demonstration, a calendar program, and a couple of other programs. The clock is the program most used on the disk with Amazing and the new fonts a close second.

Unfortunately, once again the space is the limiting factor in this column. There is much more to come on the Mac in next month's column, including more releases from the Public Domain Exchange and other sources.

Soon to come: Kermit, Icon editors, utilities galore, and more. Write to the Public Domain Exchange for its catalog (it costs $1). The group can be reached at 673 Hermitage La., San Jose, Calif. 95134 (408) 942-0309.

# Periscope ... A Direct Hit!

*"A great debugger ... If you've been frustrated by programs that don't behave the same for the debugger as when running alone, or that crash altogether, you should check out Periscope ..."*

*PC REPORT, Boston Computer Society*

*"It works, and works well!! In the first day of use I finished up two weeks of problems!! Really excellent!!!"*

*Peter Loats, Periscope User*

**Periscope's differences hit home!** Its breakout switch gives you control, even when interrupts are disabled. Periscope's unique power helps you solve the difficult problems. Debug device-drivers, memory-resident, and non-DOS programs.

**Great for straightening out confused C pointers!** Using its breakpoint power, you can stop on reads and writes to ranges of memory. Stop on registers, words, and bytes, too.

**It's tough.** Periscope's 16K RAM board is write-protected. Runaway programs can't touch crucial debugger code! Examine the system, safely, at any time: Periscope saves the interrupt vectors and buffers it uses, then restores them when done. It uses ROM BIOS calls for functions except file access, so DOS can't clobber itself.

**It gets you moving.** It's commands are similar to Debug's, so you'll master it quickly. On-line help is there when you need it. You can define the windows you want; you can design easy-to-read memory displays. Periscope supports C, Assembler, BASIC, and Pascal. Comprehensive symbol support gives you a roadmap through memory tying back to your source.

**It's risk-free.** Periscope is backed up by a 30-day, money-back guarantee! The board is warranted for a year. Technical Support? You get the best there is ... direct from Brett Salter, Periscope's author!

**It requires.** An IBM PC, XT, AT, Compaq, or close compatible; PC-DOS, 64K RAM; a disk drive and an 80-column monitor. With two monitors, Periscope's screen is displayed on the monitor not in use. With one monitor, a keystroke switches screens.

**It's power you can afford.** At $295, you can afford Periscope's professional power. The system includes the memory board, breakout switch, debugger software, manual, and quick reference card.

**Order now!** Call (800) 421-5300, ext. R96, 24 hours a day to order with MasterCard/VISA. Demo disk is $5. Questions? Please call (404) 256-3860.

Get your programs up and running;

**UP PERISCOPE!**

Data Base Decisions/14 Bonnie Lane Atlanta, GA 30328

CIRCLE 30 ON READER SERVICE CARD

INTRODUCING THE MODULA-2 SOFTWARE DEVELOPMENT SYSTEM

# The first complete programming environment brings the industry to an all-time low.

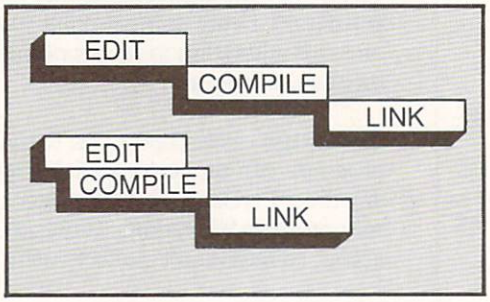
## $80.88

Modula-2 has been hailed as the programming language of the future. Its modular design and built-in error-control features make programming more efficient than ever.

And now there's a system that makes programming more affordable than ever. Interface Technologies' Modula-2 Software Development System (M2SDS).

**EASY TO LEARN. EFFICIENT TO USE.** M2SDS features a "syntax-directed" editor that makes programming easy for beginners to learn. And faster for professionals to use.

With our editor you can enter full statements with a single keystroke. And save up to 90% on typing time. It also gives on-line help in correcting undefined variables and syntax errors—which saves even more time.



*It not only has a faster compiler, it also saves time by compiling while you edit.*

Multiple editor windows let you refer to one file while you edit another. That's one more way M2SDS adds hours of more creative, more productive time to your day.

*Work faster and easier with multiple editor windows.*

EDIT TWO FILES
IN TWO WINDOWS AT ONCE

**TURN "WAIT TIME" INTO "WORK TIME."** When there's no time like real-time, you can count on the M2SDS compiler. Up to 100 lines of Modula-2 text can be turned into native machine code in less than five seconds.

To create programs using your computer's full capacity, there are 18 library modules. And unlike other low-priced compilers, M2SDS has a linker that assembles the components of your program. Automatically.

**BREAKTHROUGH TECHNOLOGY. BREAKTHROUGH PRICE.** M2SDS works with IBM® PC, XT, AT or any other 100% compatible computer. Any programs you develop, you own. And M2SDS is non-copy protected.

For just $80.88, M2SDS is the complete programming environment. Including editor, compiler, linker, library modules, 8087 support and more.

Or choose the expanded, fully upgradeable SDS-XP for just $249. Later you can add a debugger, foreign object import module and tool box for even more programming capability. And efficiency.

So whether you're a professional looking for a faster way to program, or a novice looking for an easier way to learn, there's a Modula-2 Software Development System just for you.

Call us today for more information or to order your M2SDS. Find out how our new low in system pricing can put your programming efficiency at an all-time high.

*WE ACCEPT CHECKS, MASTERCARD, VISA AND AMERICAN EXPRESS. Price does not include shipping and handling. Texas residents add 6.125% Sales Tax. International orders add $30.*

INTERFACE TECHNOLOGIES
*3336 Richmond, Suite 200, Houston, TX 77098*

GET MORE PROGRAMMING EFFICIENCY IN A SYSTEM THAT COSTS LESS. IN TEXAS, CALL (713) 523-8422.

## CALL 1-800-922-9049

CIRCLE 56 ON READER SERVICE CARD

# Programming Philosophy:

## Interviews with Donald Knuth and Niklaus Wirth

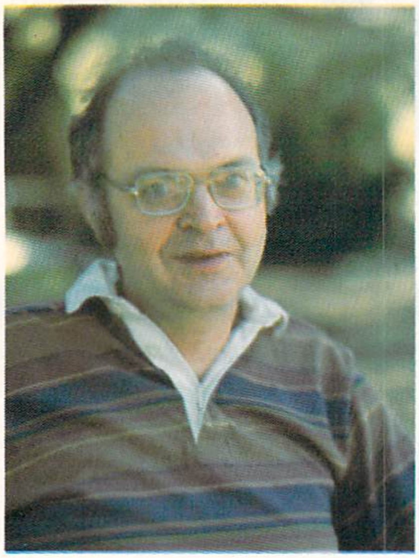
**By Ken Takara**

**This article examines the attitudes of two computer scientists—Niklaus Wirth and Donald Knuth—whose personal philosophies have profoundly affected the nature of programming.**

**Donald Knuth, professor at Stanford University, Palo Alto, Calif., is renowned for his three volumes of *The Art of Computer Programming,* a landmark in formalizing the discipline of computer science. His views of programming as an art have been expressed in numerous articles in *Communications of the Association for Computing Machinery* and in his 1974 ACM Turing Award lecture, "Computer Programming as an Art."**

**Niklaus Wirth, currently on sabbatical at the Xerox Palo Alto Research Center, Palo Alto, Calif., from the Swiss Federal Institute of Technology, is creator of the Euler, ALGOL-W, Modula, and Pascal programming languages. Wirth is equally well-known for his strong advocacy of structured programming. He was awarded the prestigious ACM Turing Award in 1984.**

**Both men are professors, which is reflected in their common concern for programming clarity. Beyond that, Knuth's background is in mathematics while Wirth is trained as an engineer. These divergent backgrounds reflect the nature of computing—it is an amalgam of mathematics, engineering, and various other disciplines. The mathematician brings theory, proof, and formal rigor. The engineer brings practicality and purpose. Or is this really so?**



**Good programming is such an inherently creative act that it's impossible to keep it to an assembly line process.**

**CL:** Your books are called *The Art of Computer Programming,* yet the discipline is called computer science.

**Knuth:** Calling it Art was my first idea for the title. I later decided to change it to *The Analysis of Algorithms,* but the publishers wouldn't let me because they said it would never sell. I'm glad they talked me out of making a change.

Art, in one of its main senses, is something human beings can do, but we don't know how to automate. Science is something we understand well enough to formalize and mechanize. As science advances, we know more about the world. And we're able to automate more.

When someone says, "Let's convert an art into a science," he's saying, "Let's let the science catch up to where the art is, to what human beings can do." What happens is, as science advances, art also advances, but even further. When we learn more science, then our human abilities jump forward too. Then there's more for science to catch up with. I hope science never overtakes art; I can't imagine that happening.

But for me, I really like the meaning of the word in the sense of fine art. That is, it's possible to write grand programs.

**CL:** How does your paper on literate programming fit in?

**Knuth:** I've been stressing the notion of style in programming. You can have large programs that are beautiful or small programs that are beautiful. You can have programs analogous to poems or novels. In that paper, I'm trying to emphasize that the best way to program, I believe now, is really to concentrate on explaining to a *person* what the computer is supposed to do rather than explaining to a *machine* what it's supposed to do.

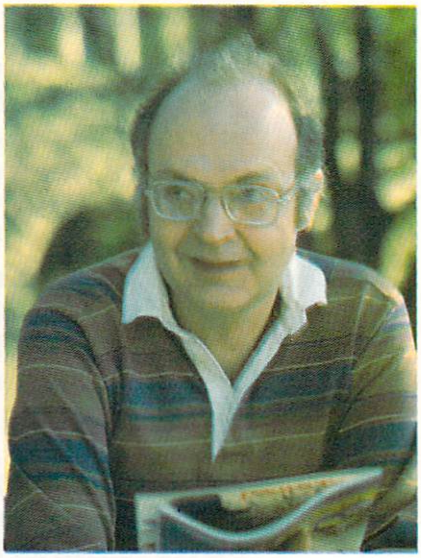
That's my new hobbyhorse, to say that people should think about the communication of the program while writing it. This makes it easier to write. The surprising thing is, that even though I'm writing programs that are better documented, it's taking me less time to write the program. I'm not losing time by taking this extra step. It's because I make fewer mistakes when I put myself in teaching mode to another human being. It's a discipline that keeps me from making errors I would make if I was hacking up something just for the computer, not really trying to explain how it works.

I start with an idea. Then I start trying to explain it to a hypothetical person.

For example, last night I needed a data structure to represent a hardware circuit for a sequential circuit. And I knew what kinds of operations I would be doing—some AI-type heuristic searches. I got an idea about a suitable structure, and before I wrote any code, I started writing down "this data structure is going to contain gates represented in the following way, these are the inputs, these are the outputs," and so on. By the time I was through explaining in English what the data structures would be and how they would work, it was almost trivial to write the code and get it right.

If I started the other way, if I had started by playing around with scraps of code and so on, I would have floundered around a while, and the program wouldn't have worked. So I start out with an English description of the data structure and with a small example. Then I write a procedure that prints out the data structure in symbolic form. I get a program that will read the structure and print it out. This program is also an example of how to use the structure. Then I write a routine that recycles or initializes the data and so on. Pretty soon the program is done.

My program is contributing to the documentation too. The best way to document a program is the way science writers have done for a long time—by saying things twice in complementary ways. You say something informally and then formally. The formal reinforces the informal because it makes things precise, and the informal reinforces the formal because it gives you handles on how to store it in your brain.



**I think future languages should be a combination of a programming language and a document language.**

(*Continued on p. 28*)

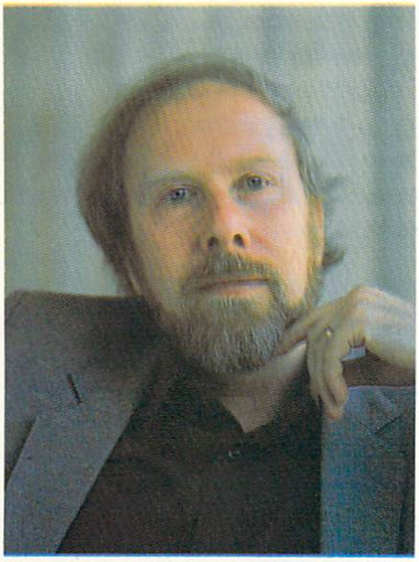
# Wirth

**CL:** Does Pascal reflect your own personal approach to programming?

**Wirth:** Certainly. And it also reflects my particular field of work. I guess most common programming languages have their own field of application where they are good.

For example, BASIC was developed at Dartmouth College many years ago, mainly with the purpose of teaching introductory programming. I think it is pretty good at doing that to a certain degree, although it lacks certain concepts that are quite fundamental.

C.A.R. Hoare compared programming in BASIC to playing the piano with two fingers. You make very fast progress for the first very simple tunes, but if you have to go further, it becomes difficult. Now, many people who learn programming don't have to go further than just playing a few simple tunes. Also, there are many areas where once you solve the simple problem, you're done, and you forget about the problem.



**Ada is an uneconomical language. It throws too many things at you . . . C is becoming popular because UNIX is written in C . . . But I don't think it's a step forward.**

My interest, however, is systems that are going to be used by many people every day for many years, like compilers or operating systems or text editors. These are complicated systems. It pays to have a good tool tailored to design complex systems in a structured way.

**CL:** Everybody talks about "structured" nowadays. Is this really the best way to approach programming?

**Wirth:** It would be presumptuous to give some kind of globally valid rule for programming style. Programming is much too diverse a field to be condensed to some set of dogma. But, of course, there are principles that have proven to be superior to others.

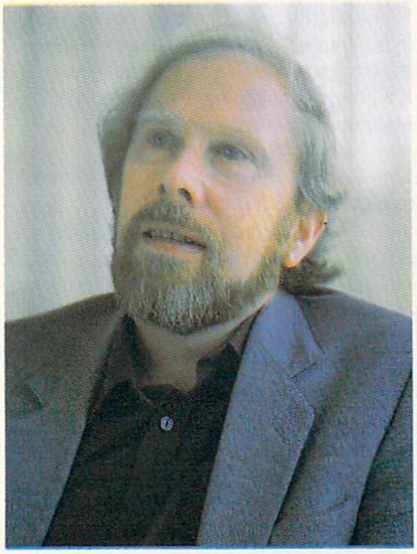
Basically, the idea of structuring a program is to mirror the structure of the algorithm by the structure of the program text. We must structure an algorithm in order to understand it. We adhere meaning to certain parts of the structure. The structure holds the individual parts together. If we have an algorithm that is 10 pages long, we can't understand it unless we know how to decompose it into manageable portions.

**CL:** To some programmers, the linear approach seems to make more sense. The business of structuring seems restrictive.

**Wirth:** Of course, it's nice to have as much liberty as possible to express what we really want to express. But when designing and inventing new programs, it's a good thing to be able to rely on proven guidelines to help us keep our thoughts ordered. I mean, it is well known that we make too many mistakes otherwise. That's really the whole game of it.

Every algorithm has an inherent structure. Maybe it has nested loops. A compiler translates them into a string of bits. The machine is unaware of the structure of the original text. It just sees a mass of bits. But we, the humans, need the textual structure to be able to understand the algorithms.

If I may state one fairly general thing—I find that most people think of the results of their programming efforts as something that is just being interpreted by a machine. Once the machine works correctly, they think the job is done. I think this is a terribly wrong attitude because the machine, in delivering one correct result, provides no guarantee that the next time, with different input parameters, the result will be correct again.



**For a vast majority of programming activity, creativity in the higher sense of the word is not an essential ingredient.**

What I mean to say is that programs must be understood by humans. Programming should be an act directed not at the machines, but at humans. At the very least, one person besides the author should understand what it does. Every program should be written such that you can show it to your colleagues and know they will be able to understand and appreciate it.

**CL:** Donald Knuth is involved with something called "literate" programming, which deals with programming so that other people can read what you wrote.

*(Continued on p. 29)*

The same thing goes for programming style, where you alternate between the informal comments and actual fragments of a program. I can hack together working programs fast, much faster than I could years ago. And I used to think I was good then. I think future languages should be a combination of a programming language and a document language. So you have a part where your text is just intended for paragraphs or math formulas and then you go into pieces of program that correspond to what you just said informally.

**CL:** You've made some comments about programming beauty and elegance.

**Knuth:** You know, life is hard. There are many un-beautiful things in the world. And the more beautiful things we have, the more tolerable it is. It's important to create beautiful tools.

We see now that a lot of the software is really pleasant for people to use day after day. The hard thing is to make something that will grow with the user—something that is easy to learn, but it's not a hassle six months later when the user has gone to a higher level of understanding.

For example, these mouse-oriented things. If you're a good typist, it's easier to hit the return key a few times to move down three lines than to pick up the mouse, move the cursor three lines, put it down, and so on. But the first week, you may really appreciate that mouse. Well, is there some way to have a system that continually grows? You can change the usage of it so that you can get better using it.

**CL:** Like a program that has plenty of help screens but where you can turn them off when you no longer need them?

**Knuth:** Yes. In the simple case, you have these game programs that ask if you want instructions. A tool, if it's really powerful, can change the user. The user becomes a different person after using it for a month and still another person after using it for a year.

This means that it's really inevitable to have "cults" among computer users—cults in the sense that these people have used a certain powerful tool for a year. Only other members who have been in the same position can understand them. And this separates them from the rest of the world.

WordStar users, for example, or LISP users, APL users—every language user. I used to think it was a menace to have cults. Then, I realized, it's inevitable. I don't know if you would want to spread them too much, but if a tool is good enough, it's got to be good enough to have a cult built around it. It means that a person can do higher level things that make sense only after they've internalized a lot of how they use this tool. I now see that cults are unavoidable and are valuable for their members.

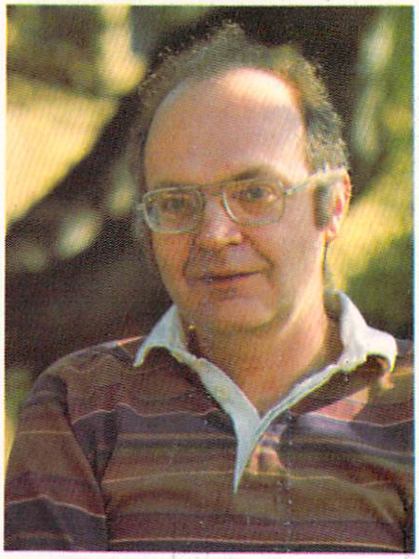
**CL:** You mentioned programming as a means of communicating to people rather than machines. There seems to be a running battle between people who think programming should be creative and original and those who want clarity and maintainability.

**Knuth:** Well, originality is important. If your only idea is to be highly original, but nobody can understand what you're doing, then you may score high on the originality scale but very low on the utility scale.

If maintainable means it has to be inefficient and full of banalities, it's because the programmers don't make use of the tools. If you have to do sequential searches instead of binary searches because binary searches are too sophisticated, then you have to limit yourself to things that require no knowledge to verify correct. You have only maintainability but no usability.

For example, Jim Morris invented a new way to find patterns in text. It was better than the obvious way because it didn't have to back up. Otherwise you would have had to worry about negative buffering. It was based on a nonobvious theory that contructs tables to remember how you got where you are; it's called the Knuth/Morris/Pratt Algorithm of Searching now.

Anyway, it's based on some magic tables, and Jim needed some mathematics to justify that it works. After he put it into this system at Berkeley, six months later, some maintainer came along, couldn't understand it, and ripped it all out and went back to an obvious algorithm. It was much slower, but at least the guy understood it.



**[Wirth] passes a law against a part of the language, while I tell people why it's bad and how to use it properly. For me, structure is a state of mind rather than a set of rules.**

Now, I don't want everybody to write programs that take a Ph.D. to understand. Lots of theoretical things have been developed that should rarely be used in real programs. The benefits are often there only if we push things to the limits. The value of these theories is that it gives people more insight into the whole structure. The more theory you know, the easier it is for you to understand the few things that you really do use in a program.

People need this theoretical bent to

*(Continued on p. 30)*

# Wirth

**Wirth:** I have always admired his work. He perhaps emphasizes somewhat less the urgency of structuring than I do. But Knuth is an outstanding man. He has a very good mind. Perhaps he can follow an algorithm without bothering to structure it so explicitly and carefully. He can afford to be a little more sloppy, if I may say so.

He just recently showed me what he did with his WEB program, integrating text processing with programming. I think it's an interesting approach. It's hard for me to comment. I haven't developed a feeling for it, not having worked with it. What occurred to me, however, was that it is a programming style a book author would invent because he is trying to describe algorithms in books. So there he had it right away. He can refer back and forth.

Now I'm not so terribly sure that's such an awfully good idea—this cross referencing. Computers are good at it, but people have some trouble. Of course, essentially we use the same method. In my books I do the same thing.

**CL:** How do you approach teaching programming?

**Wirth:** I must say, teaching by good example is so far the only effective method I have found. Here, of course, I'm in contradiction with our friend, E.W. Dijkstra, who says you must teach from general rules, totally avoiding examples. In my experience, particularly in engineering schools, it is better to start with particulars that motivate the student and then move toward the general. Dijkstra, trained as a mathematician, thinks differently.

**CL:** One of the ongoing arguments in programming concerns programmer creativity. Some people think programmers should be considered artists. Others disagree.

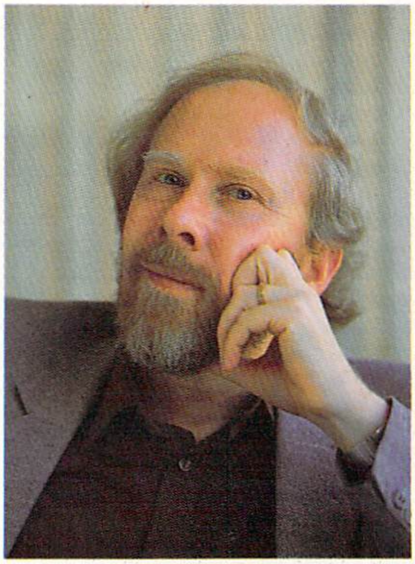
**Wirth:** Let's face one thing about creativity. It's probably much less important than we usually would like to believe. In 99% of the programs that programmers write, creativity has very little place, maybe 1%. There are very few complicated data structures; there are very few intricate algorithms. And if a programmer requires a sophisticated algorithm, he takes it from his collection of subroutines.

The major part is framework—the box it fits in. Think of an extreme example such as business data processing. You have sorting, you have searching, you have some arithmetic. You seldom have much more sophisticated things.

I don't think the use of structured languages presents any danger of reducing originality. To the contrary, a well-structured language is beneficial because it helps you put these collected algorithms into a nicer framework.

You develop a very sophisticated algorithm, like string searching, for example. Or a sophisticated method of finding eigenvalues in a matrix. I don't think the question of programming languages enters at all in the way you formulate the algorithm.

The art in engineering is not so much to make something very complicated. The art is to make a complicated problem simpler. When you develop a program, it's much harder to devise a simple solution than complicated ones. Unfortunately, our computers are terribly uncritical. They swallow anything.



**Knuth is an outstanding man . . . Perhaps he can follow an algorithm without bothering to structure it so explicitly and carefully.**

Of course, so many people have argued about this. What I mean to say is that for a vast majority of programming activity, creativity in the higher sense of the word is not an essential ingredient. The interesting kind of programming is exactly the kind that requires some amount of creativity, but for daily work, that isn't true.

Creating the first spreadsheet—now, that was highly creative. But there are not too many people who do such creative work with such imagination.

**CL:** One of the things about Pascal and Modula are their simplicity, which separate them from languages like PL/I or Ada. Is this part of your own philosophy also?

**Wirth:** Particularly as a teacher you learn to appreciate simplicity. I do not mean simplemindedness. Some people think it's the same thing. It's not. But it's nice to be able to teach clear, straight principles which you can freely combine when making logical deductions. Then the students can find their own suitable combinations where they're appropriate.

Ada set out with the same goal, I'm sure. But of course, already the requirements were baroque—very complicated and by no means without contradictions. Considering these requirements, the designers actually did a remarkably good job. PL/I or ALGOL-68 were definitely worse in this respect.

ALGOL-68 was at least consistent. That brings me to another problem. Ada is very complex. This requires complex compilers and big computers. And some people say this is going to be very expensive.

But I am afraid there is another thing which is much, much more expensive. And that is the time it takes for future generations of programmers to understand it

*(Continued on p. 31)*

extend their range. Take lists, for example. If a person knows some of the more complex theories about lists, he might never use those theories in a program, but his programming of even simple lists will be much more fluent.

I suppose it's easier to manage a group of people if you know exactly how they're supposed to do something in advance—if they're just supposed to knock something together following a very predictable set of constraints. But good programming is such an inherently intellectually creative act that it's impossible to keep it to an assembly line process. It's impossible for me to imagine managing a hundred programmers.

**CL:** What do you think are the important aspects of programming?

**Knuth:** Elegance comes in many forms. Sometimes I like to have a program that fits in a small number of bytes, and that's the criterion. Some of these game programs, both in arcades and home computers, are incredible works of art. They not only create something that responds to the human psyche, but they've also packed it into a small number of bytes.

**I believe that structured programming is a very integral part of my own style. But I still think that when I use a GOTO statement, I have an abstract meaning of what that GOTO statement is.**

Those are highly creative things. Maybe unmaintainable, but beautiful for their purpose. But other things are beautiful because they are maintainable or portable. I wouldn't stress one criterion over another.

**CL:** Niklaus Wirth says you place less emphasis on structuring code.

**Knuth:** I'm not sure about that. His Modula language doesn't have any *GOTO* statement in it. If we're talking about structure at the quantitative level instead of the qualitative level, I'm definitely different. But in more important matters we are in full agreement.

He passes a law against a part of the language, while I tell people why it's bad and how to use it properly. If they choose to use it, I don't make it impossible for them to do so. I make sure they know that they have a reason for what they're doing. So, in that sense, we have a different approach.

For me, structure is a state of mind rather than a set of rules. For example, suppose somebody said they want zero population growth. That's sort of a rule thing, "the population won't grow," instead of a state-of-mind thing, "we want a certain quality of life." And if there was a way to have the same quality of life with more population, that would be OK. To make a rule saying "ZPG" because you're hoping that this gives a better quality of life is like saying "zero *GOTO* statements," because you're hoping that this gives you better structure.

Klaus and I visited recently, and I looked up some examples where I had used *GOTO* statements in my $\TeX$ program. Almost all of them were easy to change into Modula statements, but there was one place where Modula just couldn't handle it at all. And it was a perfectly reasonable program, as both of us agreed. And Klaus said, "Well, you can't have everything. Every once in a while, you have to write a little more program in order to avoid the use of *GOTO*s." Well, that's where we disagree.

Here's what the issue was: imagine a *CASE* statement that has 10 parts to it, of which the first three parts are similar except that the first part says, "Do something and then go to a common ending;" the second part does another thing and goes to the same common ending; the third part does a third thing and goes to the common ending; the fourth part skips the common ending. And so on.

This is very common in text processing, where I had several similar cases that I want to handle at high speed. If I reprogrammed it in Modula, I would have to copy the code for the common ending because the Modula language allows you an easy way to split up into several directions but not to collapse some of the directions together.

So my philosophy about structuring is that I use *GOTO* statements in a limited way. Any time I do so, I have an abstract concept so that I know what the *GOTO* statement means. That way, I think I avoid the problems of *GOTO* statements. I've kept track of all the bugs in this $\TeX$ program over the last few years; with thousands and thousands of users, I get a lot of bug reports. Naturally I'm finding more and more subtle ones. Out of 500 bugs, something like seven or eight were due to *GOTO* statements.

**I think one of the most important things for people to learn is the ability to flash rapidly from the high level to the low level and back . . . I think that's what distinguishes the really good programmers from the ordinary ones.**

This proves that *GOTO* statements are indeed harmful. But that only covers two percent of the errors! So there are other things that are harmful too, and if we were to get rid of each thing that was harmful, we would be left without a programming language at all.

But Klaus and I do have the same point of view, that a person should understand the structure of his program by having a high-level view of it. The danger of an unstructured program is where the meaning is distributed here and there, and it's impossible to take any part of it and get a clear idea of what that part does. You

*(Continued on p. 32)*

# Wirth

**I don't think the use of structured languages presents any danger of reducing originality. To the contrary, a well-structured language is beneficial because it helps you put these collected algorithms into a nicer framework.**

all. I believe, particularly in our field, you should thoroughly understand the tool you are using. How else can you hope to understand the program?

So it is by this fact also that Ada is an uneconomical language. It throws too many things at you. I don't think you can just learn a third of Ada and be fine. There are places where you tread on one of these spots which you haven't learned about, and it backfires on you.

**CL:** Of course, some people claim that having many features is an asset. All the tools you may ever need are there. You don't have to build them all yourself, as you do in Pascal or Forth.

**Wirth:** One of the advantages of Modula over Pascal is the ability to declare operators enclosed in modules. You can compile them separately and stick them into your library. Now you may have a dozen string packages, for example, and you choose the one that is most suitable to your purpose.

It is essential to distinguish between the facilities built into the language and those you can construct. The better a language is, the more you can construct and the less needs to be built-in. Ada potentially has the advantage of being backed by a powerful and well-endowed organization employing thousands of programmers to generate rich libraries. Modula relies on the work of "volunteers," as did Pascal in the beginning.

**CL:** I know of a fellow who likes COBOL because it provides him with all sorts of features, including a *sort* command. And some PL/I programmers I know of wouldn't use anything that provides fewer utilities.

**Wirth:** First of all, if this fellow is so happy with COBOL, then by all means leave him with COBOL. On the other hand, other systems may have such libraries too. But I think it is essential that a language not be burdened with complicated operations like sorting. Sorting is expressible as a program in terms of simple operations, such as in Modula. You don't need a *sort* command built in. Otherwise there's no end to it.

A similar case is input/output. There is no I/O in a strict language such as Modula. But there are sufficient means for people who can build basic drivers for devices or build routines for number conversion, formatting, etc.

We all use these operations in every program. They are stored as subroutines in the library.

**CL:** If everyone builds their own, don't you end up with many tools but little portability?

**Wirth:** I/O in Modula is a primary example of this controversy. But let's face it, it would be preposterous for me to define a single set of I/O routines that would be fit for the whole world to live with. And so, I said, let's build just the basic facilities.

I defined a simple set that many people find adequate, but which is evidently less than adequate for certain applications. It is anybody's own choice to stick to this set, or to use someone else's, or even to create his own. Portability has its price, and it is wise not to standardize every detail.

I haven't the feeling that I can solve all the problems of the world for all the people. I/O was certainly not one of them.

Consider Pascal's standardization. It took about six years to get that document out. It took so long because of some very awkward details that could not be agreed upon. I understand I/0 was one of them. So I would have to wait another six years for Modula. It's much better if that went out without rigid standards for that sort of thing.

**CL:** What is the advantage of strong data typing?

**Wirth:** The types are the structural template we design for our data. Type checking means you use your operators and procedures consistently with your operands. You can't add two Booleans, for example, or invert a character. By declaring types explicitly, a compiler can check a program's consistency. And this is something we need in our battle against mistakes.

**CL:** Undoubtedly, there are places where strong typing is not desirable. People sometimes go to great lengths to get around the type checking.

**Wirth:** Yes. For example, you would like some generic routine that writes data on disk. To this routine, it doesn't matter whether you regard the data as logical, arithmetic, or whatever. So you need a generic read/write routine.

In programming with data types, we use abstraction. That's the whole game of it. For example, although we may know that a character is represented by seven or eight bits, we make no use of this knowledge. To convert a lower-case letter to its capital equivalent, for example, is a basic operation. The routine that writes data to the disk belongs to a lower level of abstraction. In moving from one level to another, we need to be able to relax the data type rules.

**Particularly as a teacher you learn to appreciate simplicity. I do not mean simplemindedness. Some people think it's the same thing. It's not. But it's nice to be able to teach clear, straight principles which you can freely combine when making logical deductions.**

*(Continued on p. 33)*

might think it just happens to work when you put it all together. There's no intellectual way you can understand it as a whole, since it's all mixed up.

In that sense, I believe that structured programming is a very integral part of my own style. But I still think that when I use a *GOTO* statement, I have an abstract meaning of what that *GOTO* statement is.

**CL:** Sometimes it seems that most of programming is a matter of trying to bend the algorithm to fit the constraints or peculiarities of the language you are using. Some people prefer to use the low-level languages like Forth or C in order to get away from some of these limitations.

**Knuth:** I found that this was very true at the beginning, when structured programming became a moral code. People saw so much virtue in introducing a new Boolean variable just so that they could avoid a *GOTO* statement. While it took extra lines of code, the program was still quite safe and understandable. Most of the bad *GOTO*s were the kind a good programmer would never have written in the first place. There were other kinds that good programmers would naturally write. But the easiest thing was to abolish them. I thought it was silly. I wrote a long paper about it back in '73.

**My new hobbyhorse [is] to say that people should think about the communication of the program while writing it. This makes it easier to write.**

It's so much easier to pass a rule that abolishes something than to educate people and say, "Here's what it means when we use something." It's easier to give some sort of a quantitative rather than qualitative solution. That's where art comes in, balancing the quantitative and qualitative things.

There's another case where I've had arguments with Klaus— the question was, in a language, should you be able to compare pointer variables with each other? Say you have two pointers, could you test for Pointer $X <$ Pointer $Y$? I believe most high-level languages forbid this because they say pointers are somehow special even though they are just numbers inside the computer. The prevailing opinion among language designers is that we shouldn't give a meaning to whether one pointer is less than another.

One of the arguments is, of course, that you might have garbage collection taking place at any random time and that would change the value of all the pointers. Some garbage collection schemes would change the values so that ordering relationships would be maintained. But other garbage collection algorithms would maybe change the pointers, so sometimes the relation would be "less" and later on it would be "greater." You couldn't possibly have a decent program if this were the case.

So you can make a good argument for either side. But let me point out that if you can't compare pointers in a program, then you're forcing yourself to inefficient algorithms in certain ways. You cannot, for example, have a binary search algorithm that searches a table of pointers. You have to resort to an algorithm based on equality/inequality testing instead of less-than/greater-than testing. And it's known that the best equal/not-equal algorithms are much slower than algorithms that use comparison. This means that your language is forcing you to write a program that might be 100 times slower than one you could write in assembler. Or in Forth.

**CL:** How do you teach programming?

**Knuth:** Well, I haven't taught beginning programming for a long time. I teach the course after that. My views are very much like John Bentley's, in his book of efficient programs. I try to get students in that second programming class to realize what is going on inside the computer so that their high-level programs are based on a knowledge of what the costs are.

**Some of these programs, both in arcades and home computers, are incredible works of art. They not only create something that responds to the human psyche, but they've also packed it into a small number of bytes. Those are highly creative things. Maybe unmaintainable, but beautiful for their purpose.**

In a way, this is a handicap because it takes their minds off the applications they're programming. But in other ways, it's not really a handicap because no matter what application they have, they're going to have many ways to write it. They're going to have to choose the one that seems to them to be the most efficient, whatever their notion of efficiency is.

So I try to give them a realistic notion of efficiency rather than a language-only notion of efficiency. When they write in a language, they sometimes think that if they can express something in fewer lines of code, then they have a better solution for a problem. But it might, in fact, be the worst way to do the problem as far as the machine is concerned.

I'm basing efficiency on the real cost of what the real machine does, not on the number of characters in the program or something like that. I don't want to say that efficiency is the number one thing to have in mind, but everybody has some notion of efficiency. So in my programming course I try to give a good model for what that is. But I don't want people to be hung up on it as the first priority.

*(Continued on p. 34)*

# Wirth

**CL:** Still, people look for loopholes when they are not satisfied with the way the language handles. Sometimes they go to great lengths to get around the restrictions of the language.

**Wirth:** Yes, of course. That's the long way to avoid assembly language coding. As soon as you use a loophole, your program becomes essentially nonportable. The loophole may be fairly specific to the machine.

**CL:** Strongly typed languages provide a generic machine model. This means that you're farther from the real thing. Doesn't this limit the ability of the programmer, since he or she may not know what is really happening?

**Wirth:** Yes, it is definitely a problem. When you teach students only to learn to program in a high-level language, most of them aren't quite satisfied. They'd like to learn more about what is really going on.

**Programs must be understood by humans. At the very least, one person besides the author should understand what it does.**

For computer science students, I would say it's absolutely essential that they learn about machine structures and assembly coding. Not to become assembly language programmers, of course, but to gain insight.

A good engineer must have a good understanding of the sort of tools he's using, not just know how they look. He has to know what is underneath. But I think you should be able to produce systems where a person with an understanding of what's underneath can say, "Now that I've learned it, I have enough confidence when I program in higher level instructions that the compiler is so good it will do its job and translate them to those low-level instructions. And even if I take the greatest care, I couldn't do very much better."

Knowing what's going on underneath will help a programmer do a better job using a high-level language. For instance, if he knows how much time a procedure call takes in comparison to an addition or a multiplication, he might use more procedure calls, or fewer, depending on the situation. Of course, to a beginner it doesn't matter. But when you write a systems program or a compiler, you like to have more insight.

**CL:** Some languages like Forth or C have little type checking. The programmer is totally responsible for what happens. But in return, he gets high performance and total flexibility.

**Wirth:** I find particularly engineers like to have hands-on ability. They like to know what signals go through the bus and which registers are used for what. For them, the lower level languages are more appealing. But I'm afraid the trend is toward more and more complex systems. And that's where the use of high-level languages with lots of redundancy checking ultimately proves to be not only beneficial but indispensable.

I have also designed a language which, though I wouldn't say is similar to Forth, more or less adhered to the same philosophy. It was in 1965, when I had the task of implementing an ALGOL-W compiler on Stanford's first IBM 360. There was only an assembler available, and I didn't like it so much.

I quickly designed an intermediate language. It's called PL360. It's almost typeless, but it has program structures like *for, while,* and *if* statements. In that sense, it's quite similar to Forth or perhaps to C. It's closer to the machine with respect to data but with high-level program structures. I would myself not use it today. I feel a lot more comfortable with high-level languages offering as much checking as possible.

For instance, the Lilith machine we designed does a lot of checking. An interpreter/compiler does what it can, and those things it cannot do, we do at run time. So every array is checked for its boundaries. If the hardware is done properly, this doesn't cost much. We found that when we turn off the checking, the programs are faster by only a few percent.

But the essence of data types is that most checking can be done by the compiler and does not deteriorate efficiency at execution time.

C is similar to Forth in that respect. By the way, Dennis Ritchie thinks the same way. He also felt that we should go to languages that are more structured and with more redundancy. C is becoming popular because UNIX is written in C. Better learn C when you're using UNIX! But I don't think it's a step forward.

A bad language is really terrible. A language is something you use to communicate with. In programming, we use the term language because of some work dealing with methods of describing the notation in formal terms, like Noam Chomsky proposed to do with so-called natural languages. Of course, he failed considerably, but that's where we get programming "languages." I would much prefer programming "notation," but what has happened is irreversible.

**CL:** What are the qualities of a good programmer? Are there creative individuals found in this field?

**If somebody does a good job, has learned the techniques, applies them very well, and has learned to distinguish between important and less important things, that's already the Ph.D. level. There are a few left who have all that, plus imagination and creativity. They are probably the ones you would call the artists.**

*(Continued on p. 35)*

**CL:** How about programming elegance?

**Knuth:** Right now I'm writing programs in Metafont, where I have a choice between two styles. I can write code that makes the machine work harder yet it explains my intentions better. Or I can write a program that would be much harder to read but that would run much faster. In this case, I'm trying to make these programs so that somebody can read them and modify the letters I'm designing. Even though it's going to take more computer time to generate the fonts, each font generation is not going to be done every day. So I'm putting the emphasis in this case on readability.

But if I'm writing the Metafont compiler itself and confronted by the choice between readability and efficiency, I would choose efficiency; and I would add to my documentation, explaining why the program does what it does. I would have long documentation for a more inscrutable program. But I won't do that for a program where the reader is more important than the execution.

When I taught introductory programming, I tried to emphasize there, again, that people would have some kind of a model of what the machine was like underneath as well as the algebraic language itself. I found it was more effective if I started by giving the people a little idea about what registers and such things were like.

**We see now that a lot of the software is really pleasant for people to use day after day. The hard thing is to make something that will grow with the user—something that is easy to learn, but it's not a hassle six months later when the user has gone to a higher level of understanding.**

**CL:** What languages do you use when you teach?

**Knuth:** I work with assembler when I teach my second year class and then a high-level language, usually Pascal. But I combine the two. I say, "Here's what the differences are between them." I think one of the most important things for people to learn is the ability to flash rapidly from the high level to the low level and back. That's what I tend to emphasize. I think that's what distinguishes the really good programmers from the ordinary ones—the ability to shift levels quickly.

**CL:** Your attitudes seem closer to that of an engineer than a mathematician.

**Knuth:** Few people realize the gap between a computer scientist's organization of knowledge and a mathematician's organization of knowledge. They think that they're both the same.

There have been a bunch of books lately by some mathematicians who are trying to exposit computer science the way it "ought to be done." And I know I couldn't have written one sentence of those books. Every sentence is, somehow, not a sentence that a computer scientist would write. And most of my colleagues share the same way of thinking.

That's why we're in this department. That's what makes us computer scientists. We have a variety of talents, but the main thing we do is characterized by a way of thinking that's reflected in the kinds of analogies that are easier for us to understand.

On the days I'm a mathematician, I know I'm a different person. I wrote this book called *Surreal Numbers.* That's the mathematical me writing. Nothing about computers in there. The things I know about programming and languages and so on are not used at all. There's another part of me that's very unmathematical, very algorithm oriented, and that part of me wrote some parts of $\TeX$ and Metafont.

**Every once in a while I get wonderful opportunities where I can be both [a mathematician and a computer scientist] at once, and there I feel I'm really doing what I was sent to earth to do. There I know that because I've got this mathematical ability, I can do something more for programming. It's a feeling I think is one of life's greatest pleasures.**

Every once in a while I get wonderful opportunities where I can be both at once, and there I feel I'm really doing what I was sent to earth to do. There I know that because I've got this mathematical ability, I can do something more for programming. It's a feeling that I think is one of life's greatest pleasures. You don't have to operate in only one mode all the time. Everybody has chances to build bridges between disciplines, based on a unique combination of talents. ■

**Wirth:** I had a discussion recently with one of my colleagues about what should be required of our students. The feeling was this: at the B.S. level one should expect that a student knows how to apply techniques that were taught and that he does this conscientiously. At the M.S. level, he is expected to distinguish between important and less important things— between essentials and details. He should have a taste for solid design.

Even at the Ph.D. level we see an appalling lack of this ability, actually. If somebody does a good job, has learned the techniques, applies them very well, *and* has learned to distinguish between important and less important things, that's already the Ph.D. level.

And then, there are a few left who have all that, plus imagination and creativity. They're really the exceptions. They are probably the ones you would call the artists. The really exceptional person needs both. He must have the scientific background; he must know his tools and have the ability to analyze situations and concepts. He must apply the tools accordingly, and if no adequate ones are available, create his own. And generating something new, that's asking a lot.

**What's definitely bad is the guy who thinks he's terribly creative but all he creates is shoddy work. Maybe he can inspire others to do it well, but my experience is that those who are really creative are often those who also do good, solid work.**

What's definitely bad is the guy who thinks he's terribly creative but all he creates is shoddy work. Maybe he can inspire others to do it well, but my experience is that those who are really creative are often those who also do good, solid work.

The rare person who has the ability to extend the boundaries by creatively and effectively using his tools is the true artist. Every true artist first masters the techniques needed for his art and abhors shoddy workmanship. ■

*Ken Takara received a B.S. in computer science from the Univ. of California at Santa Barbara in 1981. He has been working professionally as a programmer since 1978.*

GROWING OLD?
...waiting for C programs to compile and link?

Use C-terp
the complete C interpreter

**This is the product you've been waiting (and waiting) for!**

Increase your productivity and avoid agonizing waits. Get instant feedback of your C programs for debugging and rapid prototyping. Then use your compiler for what it does best...compiling efficient code ...slowly.

**C-terp Features**

- Full K&R C (no compromises)
- Complete built-in screen editor-- no half-way house, this editor has everything you need such as multi-files, inter-file move and copy, global searching, auto-indent, tab control, and much more.
- Fast--Linking and semi-compilation are breath-takingly fast. (From edit to run completion in a fraction of a second for small programs.)
- Convenient--Compiling and running are only a key-stroke or two away. Errors direct you back to the editor with the cursor set to the trouble spot.
- Compiler Compatible--You can access functions and externals compiled with C86 or Lattice C or assembly language. Utilize your existing libraries unchanged!
- Complete Multiple Module Support-- Instant global searches, auto-compile everything that's changed, etc.
- Many more features including batch mode and symbolic debugging.
- Runs on IBM PC, DOS 2.x, 192K and up
- **Price: $300.00 (Demo $45.00) MC, VISA**

*Price of demo includes documentation and shipping within U.S. PA residents add 6% sales tax. Specify C86 or Lattice version.*

GIMPEL SOFTWARE
3207 Hogarth Lane • Collegeville, PA 19426
(215) 584-4261

*Trademarks: C86 *(Computer Innovations)*, Lattice *(Lattice Inc.)*, IBM *(IBM Corp.)*, C-terp *(Gimpel Software)*

# HCR/PASCAL, WIRTH ITS WEIGHT IN C

**PASCAL** Originally designed by Niklaus Wirth, is now available for a wide range of UNIX™ processors. HCR/PASCAL conforms closely to industry standards, passes all conformance tests in the PASCAL Validation Suite. Supports multiple module programs, a dynamic string package, and direct random file access.

**C** is the standard language of UNIX, HCR/PASCAL is written in C and translates PASCAL into C producing efficient optimized code. This approach allows direct interaction with the UNIX environment and offers a high degree of portability.

**UNIX** is a powerful yet flexible operating system environment. HCR/PASCAL is available today on a diverse range of UNIX hardware: AT&T 3B™ series, the NCR Tower,™ DEC PDP-11/VAX,™ and others. HCR has a growing line of UNIX software including business applications. We back up all our software with full support. To find out how we can put HCR/PASCAL, C, and UNIX together for you, call or write:

Human
Computing
Resources
Corporation

HCR

10 St. Mary Street, Toronto, Ontario, Canada M4Y 1P9 (416) 922-1937

See us at UniForum Booth #1441

UNIX is a trademark of Bell Laboratories. PDP-11, VAX, and DEC are trademarks of Digital Equipment Corporation. AT&T 3B is a trademark of American Telegraph & Telephone. NCR Tower is a trademark of NCR Corporation.

**CIRCLE 93 ON READER SERVICE CARD**

# Save Time Using BASIC Macros

**By Dwaine L. Bendorf**

Programming with macros—the concept of assigning a series of operations or functions to a simple expression—has been around a long time.

Assembly language programmers were the first to develop and use macros and macro libraries to aid in the writing of operating systems, compilers, and other large programming projects (most of which they would still be working on if they hadn't used macros!)

If you are programming in BASIC and you spend much of your time writing and debugging what seems to be the same code over and over again, or if you have ever wished for an organized method of saving and reusing complicated algorithms, or if you have read about programming in C with macros and libraries and thought "Boy! I wish BASIC had some of that power," then macro programming could be for you.

There is a catch: you can take advantage of macro programming only if you are willing to modify your program development cycle to include the use of a structured programming language and preprocessors. This may sound like a lot of trouble.

However, the time it will take you to adapt to using these tools will be measured in hours instead of the days or even weeks it would take you to learn a new language and become as efficient as you are in BASIC. This initial investment of time, effort, and added steps will provide you with big dividends in increased enjoyment and productivity.

### Implementing macros

The use of a BASIC macro language preprocessor (BMLP) along with a BASIC structured language preprocessor (BSLP) can open the domain of macros to you and your next project. These tools are available in MBASIC source form from the *COMPUTER LANGUAGE* Bulletin Board Service and the magazine's account on CompuServe. If you do not have a modem, this source plus all listings in this issue can be purchased for $5 by writing to the editors.

**Formats.** Let's first take a look at the macro expression format:

```
$macro-name param1,param2
```

A leading dollar sign ($) is used to identify a macro name and, except for spaces and tabs, must be the first character on the line. When parameters are used, they must be separated by a comma, tab, or space; when there are too many to put on one line, a double backslash ( \ \ ) may be used to continue on the next line.

The macro coding format is:

```
MACRO <macro-name>
ENDM
```

Macros are written and maintained with a text editor in one or more library text files or program source files. Macro names may contain any combination of characters except commas, tabs, and spaces, and there is no case discrimination.

The keyword *MACRO* is used to identify a macro name and the beginning of code for that macro name. The keyword *ENDM* identifies the end of code for a macro name. Within the macro code, a number enclosed with brackets ([]) identifies a parameter request. A leading semicolon (;) may be used as a comment character. Macros may use other macros as well as supporting subroutines (more on this later).

**Design.** Writing macros is easy, fun, and can be habit forming once you get the hang of it. To demonstrate, let's take the simple task of displaying text on the video monitor. The following list of operations describes this task in detail:

1. Save the current cursor location
2. Turn off and reposition the cursor
3. Clear a space the size of the text
4. Reposition the cursor
5. Display the text
6. Restore cursor location and turn it on.

BASIC code for this example might be:

```
100 XR%=CSRLIN:XC%=POS(0)
    :LOCATE,,0:LOCATE ROW,COL
    :PRINT SPACE$(LEN(TEXT$));
    :LOCATE ROW,COL
    :PRINT TEXT$;
    :LOCATE XR%,XC%,1
```

A macro expression for the preceding example is:

```
$CRT ROW,COL,TEXT$
```

Code for the *CRT* macro could be written as in Listing 1. The parameters are numbered in left to right order as they appear in the list following the *CRT* macro, for example:

```
MACRO CRT
   XR%=CSRLIN:XC%=POS(0)
   LOCATE,,0:LOCATE [1],[2],0:PRINT SPACE$(LEN([3]));
   LOCATE [1],[2],0:PRINT [3];:LOCATE XR%,XC%,1
ENDM
```

Listing 1.

[1] = ROW <parameter one>
[2] = COL <parameter two>
[3] = TEXT$ <parameter three>

Parameters can be passed using variables or literal expressions:

$CRT 12,10,''Hello World!''

**Nesting macros.** Because a macro can use other macros (nesting), the *CRT* macro can also be written using macros for the required six functions (Figure 1).

```
    1. Save the current cursor location -------+
    |  MACRO SVC                               |
    |      [1]=CSRLIN:[2]=POS(0)               |
    |  ENDM                                    |
    +------------------------------------------+

    2. Position the cursor and switch on/off --+
    |  MACRO XY                                |
    |      LOCATE [1],[2],[3]                  |
    |  ENDM                                    |
    +------------------------------------------+

    3. Clear a space the size of the text -----+
    |  MACRO CLR                               |
    |      PRINT SPACE$([1]);                  |
    |  ENDM                                    |
    +------------------------------------------+

    4. Reposition the cursor ------------------+
    |  (Use the XY macro, (2))                 |
    +------------------------------------------+

    5. Display the text -----------------------+
    |  MACRO SHO                               |
    |      PRINT [1];                          |
    |  ENDM                                    |
    +------------------------------------------+

    6. Restore cursor location, turn it on ----+
    |  (Again use the XY macro, (2))           |
    +------------------------------------------+
```

Figure 1.

Now the *CRT* macro can be written using the four new macros:

```
MACRO CRT
    $SVC XR%,XC%
    $XY ,,0
    $XY [1],[2],0
    $CLR LEN([3])
    $XY [1],[2],0
    $SHO [3]
    $XY XR%,XC%,1
ENDM
```

Writing macros for the lower level functions of the *CRT* macro provides several benefits. One is the availability of lower level macros to be used in writing other higher level macros.

## Conditional logic

*$IF-$ELSE-$END* are reserved macro keywords that provide the ability to control the inclusion or expansion of segments of code within a macro. The *CRT* macro can be enhanced with added ability to display the text in reverse video. This is easily implemented using a fourth parameter and conditional logic to control the inclusion of the *COLOR* statement (Listing 2).

**Logical operators.** The equal to (=) and not equal to (# or < >) logical operators can be used to test literal parameters for a specific value (Listing 3).

## Macros and subroutines

One of the more powerful features of macro programming is the ability to write macros that require supporting subroutines. This ability allows the use of a macro any number of times within a program, with only one inclusion of its supporting subroutine. Thus, blocks of duplicated code are eliminated.

An example is the operation of stripping leading and trailing space and tab characters from a string of text (Listing 4).

**Developing libraries.** Macros can be written within a program source file, where they can be easily tested and debugged before being committed to a library file. Library documentation is extremely important.

After writing a couple of dozen macros, it becomes a little more difficult to remember function and parameter requirements for each macro. The ability to share libraries among several programmers working on the same project makes the library documentation essential.

**Using libraries.** The keyword *LIBRARY* is used at the top of a program source file to identify each macro library to use in processing that source file. Libraries can be nested by specifying libraries within libraries.

### BASIC and preprocessing

The BASIC language, with the help of all the great add-on packages—like windowing, B-tree file access, and graphics— has become a powerful, versatile programming language.

However, BASIC has its problems and limitations, which include having to use line numbers and line number references, using .25K line lengths to overcome the absence of multiple line conditionals, and expressing a subroutine call as *GOSUB 24190* instead of *GOSUB GET-KEYBOARD-INPUT*.

Preprocessing can take you away from all that by making your HAL do all the dirty work. An environment using a structured language, macros, and libraries for your BASIC programming will let you have your cake and eat it too.

The BASIC interpreter can be utilized to test and optimize algorithms. When fin-

```
MACRO CRT
    $SVC XR%,XC%
    $XY ,,0
    $XY [1],[2],0
    $CLR LEN([3])
    $XY [1],[2],0
    $IF [4]        <--+  If fourth parameter ([4]) is provided
    COLOR 0,7         |  then the statement COLOR 0,7 will
    $END           <--+  be included. (Reverse)
    $SHO [3]
    $IF [4]        <--+  If fourth parameter ([4]) is provided
    COLOR 7,0         |  then the statement COLOR 7,0 will
    $END           <--+  be included. (Reset to Normal)
    $XY XR%,XC%,1
ENDM

An example:

$CRT 12,10,"Hello World!",REV
                            \..(fourth parameter)
(Display "Hello World!" at row 12, column 10, in reverse video.)

$CRT 12,10,"Hello World!"
                            \..(missing fourth parameter)
(Display "Hello World!" at row 12, column 10, in normal video.)
```

Listing 2.

```
$IF [1] = 1
$IF [6] # NOERROR
$IF [4] = REV
        \
        (Operators must be separated by spaces or tabs.)

CRT macro with logical operators:

MACRO CRT
    $SVC XR%,XC%
    $XY ,,0
    $XY [1],[2],0
    $CLR LEN([3])
    $XY [1],[2],0
    $IF [4] = REV  <--+  If fourth parameter ([4]) EQUALS "REV"
    COLOR 0,7         |  then the statement COLOR 0,7 will
    $END           <--+  be included. (Reverse)
    $SHO [3]
    $IF [4] # NORM <--+  If fourth parameter ([4]) NOT EQUAL
    COLOR 7,0         |  "NORM" then the statement COLOR 7,0
    $END           <--+  will be included. (Reset to Normal)
    $XY XR%,XC%,1
ENDM
```

Listing 3 *(Continued on following page).*

ished, you can write the algorithms in structured code and put them in a library along with good documentation, then use a macro to call them into your program source file any time they are needed.

If in the future you decide you need to learn another language like C, Pascal, or Modula-2, you will find it much easier because you will be thinking structure instead of spaghetti.

BMLP and BSLP are both written in what I call SSS (pronounced "S") for Symmetrically Structured Syntax code. BSLP translates SSS to standard, line numbered BASIC code. The SSS language is composed of a small, easy to remember set of keywords that provides named procedures, multiline conditionals, a *case* construct, and several loop constructs.

Those who are using SSS say it's a snap to learn. The BASIC code version of these preprocessors will run in the BASIC interpreter, but if you do any serious work with them, you will want to compile them to reduce the processing time.

*Dwaine Bendorf is founder and owner of Bendorf Associates, a custom software house in Roswell, N.M. He has been involved in hardware and software design since the advent of the first 914 flip-flop.*

```
An example:

$CRT 12,10,"Hello World!",REV
                            \..(fourth parameter)
(Display "Hello World!" at row 12, column 10, in reverse video.)

$CRT 12,10,"Hello World!",NORM
                            \..(fourth parameter)
(Display "Hello World!" at row 12, column 10, in normal video.)
```

Listing 3 *(Continued from preceding page).*

```
$STRIP TEXT$

The code for STRIP macro:--------+
MACRO STRIP                      |
    X$=[1]:Gosub _Stripit:[1]=X$ |
    $$_STRIP                     |
ENDM                             |
---------------------------------+

The code for _STRIP supporting subroutine:------------------+
MACRO _STRIP                                                |
proc _Stripit                                               |
    unless LEN(X$)<3                                        |
        X%=LEN(X$)+1                                        |
        WHILE(X%>LEN(X$) AND LEN(X$)>2)                     |
            X%=LEN(X$)                                      |
            X$=LEFT$(X$,LEN(X$)+(RIGHT$(X$,1)=" "))         |
            X$=LEFT$(X$,LEN(X$)+(ASC(RIGHT$(X$,1))=9))      |
            X$=RIGHT$(X$,LEN(X$)+(ASC(X$)=32 OR ASC(X$)=9)) |
        WEND                                                |
    endu                                                    |
endp                                                        |
ENDM                                                        |
------------------------------------------------------------+
(The above subroutine is written in the structured language
 that was referred to earlier.)

The leading double dollar sign ($$) identifies _STRIP
as a subroutine which is defined as a procedure (_Stripit),
and it will be included into a program source file only one
time. A subroutine may include and use other subroutines
and macros.
```

Listing 4.

# Introducing the MIX Editor

## (with Split Screen - both horizontal and vertical)

## A Powerful Addition To Any Programmer's Tool Box

- Full Screen Editing
- WordStar Key Layout
- Custom Key Layouts
- Terminal Configuration
- Help Files
- Backup Files

*Introductory Offer Only*

**29.95**

30 Day Money Back Guarantee

- Programmable
- Macro Commands
- Custom Setup Files
- Mnemonic Command Mode
- Multiple File Editing
- Split Screen Editing

*For PCDOS/MSDOS (2.0 and above/128K) • IBM PC/Compatibles, PC Jr., Tandy 1000/1200/2000, & others*
*For CPM80 2.2/3.0 (Z80 required/64K) • 8" SSSD, Kaypro 2/4, Osborne I SD/DD, Apple II, & others*

### Great For All Languages

A general purpose text processor, the MIX Editor is packed with features that make it useful with any language. It has auto indent for structured languages like Pascal or C. It has automatic line numbering for BASIC (255 character lines). It even has fill and justify for English.

### Terminal Configuration

A utility for defining terminal features (smart features included) allows the editor to work with any terminal. Over 30 of the most popular terminals are built-in.

### Custom Key Layouts

Commands are mapped to keys just like WordStar. If you don't like the WordStar layout, simply change it. Any key can be mapped to any command. You can also define a key to generate a string of characters, great for entering keywords.

### Split Screen

You can split the screen horizontally or vertically and edit two files simultaneously.

### Macro Commands

The MIX Editor allows a sequence of commands to be executed with a single keystroke. You can define a complete editing operation and perform it at the touch of a key.

MIX software
2116 E. Arapaho
Suite 363
Richardson, Tx 75081
(214) 783-6001

MSDOS is a trademark of Microsoft
PCDOS is a trademark of IBM
CPM80 is a trademark of Digital Research
WordStar is a trademark of MicroPro

**CIRCLE 42 ON READER SERVICE CARD**

### Custom Setup Files

Custom keyboard layouts and macro commands can be saved in setup files. You can create a different setup file for each language you use. The editor automatically configures itself using a setup file.

### Command Mode

Command mode allows any editor command to be executed by name. It is much easier to remember a command name versus a complicated key sequence. Command mode makes it easy to master the full capability of the editor. Frequently used commands can be mapped to keys. Infrequent commands can be executed by name.

### Editor Commands

The editor contains more than 100 commands. With so many commands, you might think it would be difficult to use. Not so, it is actually extremely simple to use. With command mode, the power is there if you need it, but it doesn't get in your way if you don't. Following is a list of some of the commands.

### Cursor Commands

Left/Right/Up/Down
Tab Right/Tab Left
Forward Word/Backward Word
Beginning of Line/End of Line
Scroll Up/Scroll Down
Window Up/Window Down
Scroll Left/Scroll Right
Top of File/Bottom of File

• • •

### Block Commands

Copy/Move/Delete
Read/Write
Lower Case/Upper Case
Fill/Justify
Print

### File Commands

Directory (with wild cards)
Show File/Help File
Input/Output File
Delete File/Save File

### Other Commands

Split Screen/Other Window
Find String/Replace String
Replace Global/Query Replace
Delete Line/Undelete Line
Delete Word/Undelete Word
Insert Mode/Overwrite Mode
Open Line/Join Line
Duplicate Line/Center Line
Set Tab/Clear Tab

• • •

To Order: **Call Toll Free 1-800-622-4070,** (Illinois only **1-800-942-7317)**

Mix Editor ____ $29.95 + shipping ($5 USA/$10 Foreign) Texas residents add 6% sales tax

Visa ____ MasterCard ____ Card # ____________ Exp. Date ________
COD ____ Check ____ Money Order ____ Disk Format ____________
Computer ____________ Operating System: MSDOS ____ PCDOS ____ CPM80 ____
Name ____________
Street ____________
City/State/Zip ____________
Country ____________
Phone ____________

MIX software
2116 E. Arapaho
Suite 363
Richardson, Tx 75081

Dealer Inquiries Welcome
**Call (214) 783-6001**

# TBASIC

## Featuring powerful new ANSI proposed GRAPHICS, GPIB control, and support for Tektronix, Hewlett-Packard, DEC, and IBM

**Key features of TBASIC**

- Tektronix 4050 BASIC compatible
- Supports integer data types
- Multiple statements per line
- 31 character variable names
- String, integer, and real arrays
- Selectable subscript base of 0 or 1
- SORT automatically orders array data
- SOUND generates music/sound effects
- AUTOmatic line number generation
- Abbreviated statement entry format
- Program listings "neatly" formatted
- Instant syntax checking "shows" errors
- Line editor with 7 function keys
- 20 "user definable" special function keys
- Line labels for "structured" programming
- True subprograms with local variables
- "Implied" FOR/NEXT loop types
- Double precision arithmetic (18 digits)
- 8087 math co-processor supported
- Operator types include: arithmetic, string, logical, binary, and relational
- All math functions operate on arrays
- Matrix functions: DET/IDN/INV/MPY/TRN
- PICTURE subprograms with transforms– SHIFT, SHEAR, SCALE, and ROTATE
- WINDOW and VIEWPORT capability
- PLOT LINE/POINT/AREA/TEXT/ARC/CIRCLE
- AXIS generation with major/minor tics
- Device independent I/O
- Random and Sequential file types
- USING formats for PRINT and INPUT
- IMAGE has two formats to select from
- Binary, octal, hex, & decimal conversions
- PROMPT and ALTER options for INPUT
- DEBUG mode with TRACE and BREAK
- Single Step key for manual execution
- Complete error trapping and handling
- OPTION– to customize your environment
- ASK/SET over 20 program parameters
- HELP gives on line keyword descriptions
- RENAME variables automatically
- EXCLUDE comments from programs
- MOVE/COPY program segments
- LREF/VREF cross-reference facility
- Extensible through external routines
- Translator programs for other languages

**Operating systems supported:**

MS-DOS, PC-DOS, CP/M-86,
UNIX, and VAX VMS.

**Computers Supported:**

IBM's PC/XT/AT, AT&T 6300,
IBM compatibles: Compaq, IBS,
Leading Edge, Mindset, etc.
HP-150 and the HP Integral PC,
Tektronix 4170/4100/6000,
White Chappel MG-1
DEC VAX and Rainbow

**Device Drivers Supported:**

**GPIB Controllers:**

National Instruments, IBM,
Ziatech, Capital, and Tecmar.

**Graphic Cards:**

IBM, Hercules, Tecmar, etc.

**Graphic Terminals:**

Tektronix 4100/4010/4014, Ramtek,
Seiko, Envision, Chromatics, & DEC.

**Plotters:**

Cal-comp, Hewlett-Packard,
Tektronix, & Houston Instruments.

TBASIC™

TBASIC is not another futile attempt to improve Microsoft BASIC (like the so called Better BASIC, Professional BASIC and others). TBASIC is a true heavy weight BASIC— the first one for micros— with features common to Hewlett-Packard and Tektronix engineering BASICs. And, its improved with features from proposed ANSI BASIC.

TBASIC is the single most useful language you will ever invest in. If you have already invested in another BASIC, ask about our limited offer to buy your BASIC when you buy ours.

**Call 1-801-224-6550 for more information.**
**Demo disks are available for $20.**

**CIRCLE 82 ON READER SERVICE CARD**

TransEra

TransEra Corporation
3707 North Canyon Road
Provo, Utah 84604
801-224-6550

## Solving the Towers of Hanoi puzzle

# BASIC Recursive Techniques

### By Hugh Aguilar

> *. . . Damnable iteration . . . able to corrupt a saint.*
>
> *—William Shakespeare*

Recursion is more than just a procedure that calls on itself as a subprocedure. This is true in all languages, even those toted as recursive languages.

Let's look at recursion in BASIC by using a familiar puzzle called the Towers of Hanoi. But before we discuss recursive procedures, let's examine procedures in general.

A procedure is a chunk of code that performs a specific task. It is defined in terms of a simpler, previously defined procedure, or it is invoked by a machine language call. It is used in the definition of other more complex procedures. This hierarchical method of building up a program allows a programmer to write progressively more powerful routines while concentrating on one limited task at a time.

A procedure usually requires data, called input parameters, that it uses in performing its task. The output parameters are any data the procedure may give out to the higher level procedure that called on it.

Let us say the procedure *WASH* has a value stored in the variable *GALLONS*. The subprocedure *RINSE* needs this value in its variable *HOWWET* to work properly. When it is finished, *RINSE* has a value in its variable *HOWHOT* that *WASH* needs to get into its variable *TEMPERATURE*. The BASIC line

```
LET HOWWET = GALLONS; GOSUB
     RINSE; LET TEMPERATURE =
     HOWHOT
```

does the trick.

However, this technique has several faults. Most apparently, the author of *WASH* must understand the intricacies of *RINSE*. Knowledge of *HOWWET* and *HOWHOT* is necessary to use *RINSE*. This is a problem if someone else on the team wrote *RINSE*. Also, to avoid conflict everyone on the team must have an updated list of variable names that the others have used. If *RINSE* used *GALLONS* directly instead of *HOWWET* and had a statement like

```
FILL IF GALLONS < > 0 THEN
     GOSUB ADDAGAL:
     GALLONS = GALLONS - 1:
          GOTO FILL
```

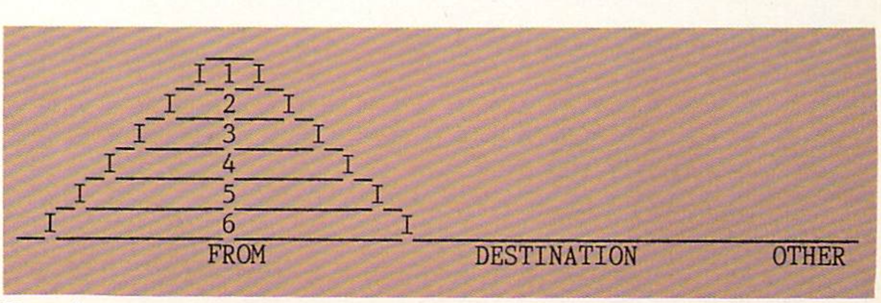
then *WASH* would bomb the next time it tried to use *GALLONS*.

Recursion is also impossible with this technique. As you will recall, recursion occurs when a procedure calls upon itself as a subprocedure. The Towers of Hanoi problem is an excellent illustration.

The task requires moving some number of disks from one peg of three to another. The disks are progressively smaller with the largest on the bottom and only one disk may be moved at a time. I urge you to cut out one-half dozen circles of corrugated cardboard and, if no one is watching, solve the puzzle on your desk.

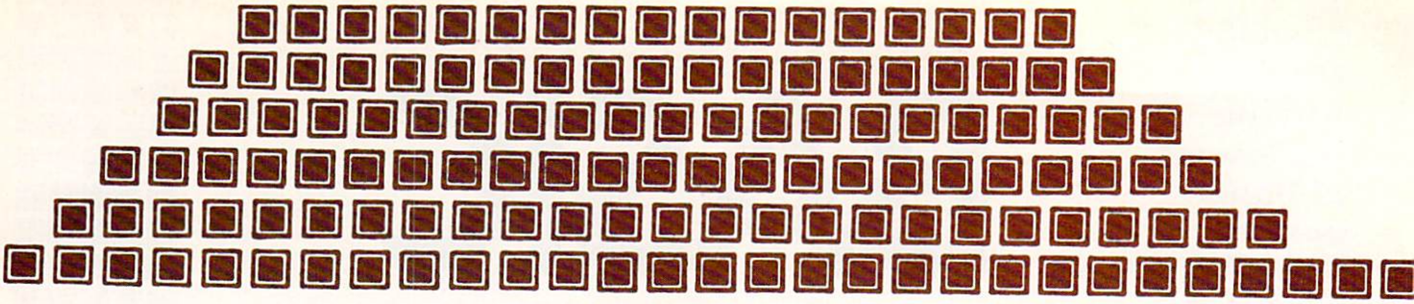
The Hanoi procedure has four input parameters: DISKS—the number of disks to be moved, FROM—the name of the peg they are on, DESTINATION—the name of the peg where they will go, and OTHER—the name of the other peg. There are no output parameters. The solution is presented in Figure 1.

Sections A.1 and A.3 require the movement of some number of disks from one peg to another. Does that sound familiar? It should; it's the Hanoi problem in miniature (one less disk). Listing 1 is a conversion of our English solution to Pascal.



```
             ___
           _I_1_I_
         _I___2___I_
       _I_____3_____I_
     _I_______4_______I_
   _I_________5_________I_
__I___________6___________I______________________________
             FROM              DESTINATION         OTHER
```

Figure 1.



As you probably discovered from your cardboard puzzle, the problem is very easy when there are only two disks to move. In that case the A.1 and A.3 sections of the solution can be done directly, as they require the movement of only one disk. However, having three disks to move presents the problem of moving two disks in the A.1 and A.3 sections.

Recursion allows for the reduction of a problem into simpler and simpler subsets of itself until a solution can be found.

Transferring data from one level to the next creates a problem. While the solution presented in Listing 2 is probably the easiest to implement, it is more of a trick than a technique. A much more sophisticated method is used in writing recursive language compilers.

A parameter stack is used to hold the input and output parameters during a recursive call. Forth programmers know all about the parameter stack, but users of other recursive languages, like Pascal, are often unaware of its existence. Stacks are LIFO (last-in-first-out) data structures.

The parameters are pushed onto the stack before every subprocedure call and pulled off upon the return of control. This assures the integrity of the variables within the context of a procedure. BASIC can be made to do this with a simple wedge.

A wedge is a method of redefining and extending certain BASIC commands and routines. It involves replacing the *CHARGOT* routine that is kept in RAM, thus allowing you to intercept each BASIC command before it is executed.

The *GOSUB* and *RETURN* commands must be extended. The *LET* command will be redefined to emulate the Pascal procedure command. Nothing is lost in this as the use of the command *LET* was only optional before. Listing 3 is the BASIC Hanoi procedure with the wedge in place.

Most BASICs store their data as in Figure 2, with the pointers *VS* (variable start), *AS* (array start), *AE* (array end), *SS* (string start), and *EM* (end of memory) kept somewhere on the zero page. String variables are intermixed with numeric variables and contain only the address of the string of charaters they represent, somewhere between *SS* and *EM*. The variables and arrays grow upward as new ones are introduced and the strings grow downward. When the two meet, the out-of-memory error occurs.

The GOSUB must store the values of the *VS*, *AS*, *AE*, *SS*, and *EM* pointers immediately after the last array. The values in the first set of parentheses (i.e., the input parameters) are calculated and put into nameless dummy variables immediately after that. *VS* is made to equal the value of *AE* + 10 (the 10 is for the 10 bytes that the pointer values occupied). This effectively hides all of the preceding data from the BASIC interpreter.

*AS* and *AE* are made to equal the address immediately following the dummy variables. The variable names in the second set of parentheses (i.e., the output parameters) are stored without any

```
PROGRAM NO.ONE (INPUT, OUTPUT);

VAR
  HEIGHT:  INTEGER;

PROCEDURE HANOI (DISKS: INTEGER;
                 FROM, DESTINATION, OTHER: CHAR);

   BEGIN
   IF DISKS > 0 THEN
      BEGIN
      HANOI (DISKS-1, FROM, OTHER, DESTINATION);
      WRITE (`Move the' DISKS, `numbered disk from ');
         WRITELN (FROM, ` to ', DESTINATION);
      READLN;
      HANOI (DISKS-1, OTHER, DESTINATION, FROM)
      END
   END;

BEGIN
WRITELN (`Height?');
READLN (HEIGHT) ;
HANOI (HEIGHT, `a', `b', `c');
WRITELN (`Finished')
END
```

Listing 1.

```
        LET DISKS   DISKS-1:  LET A$  OTHER$: LET OTHER$   DESTINATIONS$:
           LET DESTINATIONS$   A$:  GOSUB HANOI: LET DISKS    DISKS 1:
           LET A$   DESTINATIONS$:  LET DESTINATIONS$   OTHERS$:
           LET OTHERS$  A$
```

Listing 2.

values associated with them prior to the start of the strings.

*SS* and *EM* are made to equal the address immediately preceding the variable names, effectively hiding them and the strings. The result of all of this manipulation is shown in Figure 3. Then we call the *GOSUB* routine in the BASIC ROM.

The *LET* command actually only handles the input parameters and lets the *RETURN* handle the output parameters. *LET*'s job is simply to name the nameless dummy variables pointed to by *VS* with the list of names in its set of parentheses.

The *RETURN* command makes its temporary pointer equal to *VS*-10. The *VS*, *AS* and *AE* pointers are then restored to their original values pointed to by the temporary pointer. The variable names pointed to by *EM* are then looked up in the variables pointed to by *VS*. These variables are given the values in *RETURN*'s set of parentheses. *SS* and *EM* are restored to their original values, and the *RETURN* routine in the BASIC ROM is then called.

Recursion should be used when an iterative solution (i.e., one using loops) can not be found. Listing 4 is an iterative solution to the Hanoi problem. The machine understands it more easily than the recursive solution, but the iterative solution defies human comprehension. Of course, people who put a higher priority on readability than efficiency will still like the recursive solution.

Certain rules must be followed to make the use of recursion effective. Most importantly, there must be an escape clause such as the line

IF DISKS < = 0 THEN RETURN ()

The programmer must also be aware of the return stack. The *GOSUB* routine in the BASIC ROM pushes the line number and place in the line onto this stack before performing a *GOTO*. The *RETURN* routine pulls these values off the return stack.

The size of the return stack is limited and will cause problems if there are more than 23 more *GOSUB*s than *RETURN*s. This can easily occur in recursion.

I would welcome reader comments on this article. Please write me c/o *COMPUTER LANGUAGE*, 131 Townsend St., San Francisco, Calif. 94107.

*Hugh Aguilar is vice president of Deus Ex Machina Software Co. and is involved with 8-bit compilers.*

```
               VS          AS        AE          SS      EM
BASIC program, 'variables, 'arrays, 'garbage, 'strings'
```

Figure 2.

```
                                                    AE       EM
                                         VS         AS       SS
BASIC program, variables, arrays, old    ,'variables,'garbage,'strings
                                  pointer  without
                                  values   names
```

Figure 3.

```
1000 INPUT "Height"; HEIGHT
1010 GOSUB (HEIGHT, "a",  "b", "c") () 2000
1020 PRINT "Finished"
1030 END

2000 LET (DISKS, FROM$, DESTINATION$, OTHER$)
2010 IF DISKS <= 0 THEN RETURN ()
2020    GOSUB (DISKS-1, FROM$, OTHERS$, DESTINATIONS$) () 2000
2030    PRINT "Move the" DISKS "numbered disk from ";
2040       PRINT FROM$ " to " DESTINATION$
2050    FOR L = 0 TO -1 STEP -1: GET DL$: L = DL$ <> "": NEXT
2060    GOSUB (DISKS-1, OTHER$, DESTINATION$, FROM$) () 2000
2070    RETURN ()
```

Listing 3.

```
4000 DIM S(3,50): S(1,0) = 99: S(2,0) = 99: S(3,0) = 99
4010 REM 50 is an arbitrary limit in height
4020 REM 99 is just a number greater than the limit in height
4030 F = 1: 0 = 2: D = 3: INPUT "Height"; H
4040 FOR I = 1 TO H: S(1,I) = H 1 - I: NEXT
4050 H(1) = H: H(2) = 0: H(3) = 0: GOSUB 5000: PRINT "Finished": END

5000 PRINT "Move the" S(F,H(F)) "numbered disk from";
5010    PRINT CHR$(F+64) " to " CHR$(D+64): INPUT DL$
5020 H(D) = H(D)+1: S(D,H(D)) = S(F,H(F)): H(F) = H(F) -1
5030 IF H(F) = 0 AND H(0) = 0 THEN RETURN
5040 IF S(0,H(0)) < S(F,H(F)) THEN F = 0
5050 IF S(F,H(F))/2 = INT(S(F,H(F))/2) THEN D = F+1:
     IF D > 3 THEN D = D - 3
5060 IF S(F,H(F))/2 < > INT(S(F,H(F))/2) THEN D = F-1
     IF D < 1 THEN D = D + 3
5070 FOR I = 1 TO 3: IF I < > F AND I < > D THEN 0 = I
5080 NEXT: GOTO 5000
```

Listing 4.

pluto T.M.

**PLUTO—MORE THAN JUST A BASIC!**

PLUTO is a powerful, multiple user Business Basic programming environment for the IBM PC, XT, AT and the PC compatibles.

In addition, PLUTO interprets MAI/Basic Four and Science Management Corporation SMC Business Basic programs to run on microcomputers.

Powerful features of PLUTO not found in other Basics:

- Multiple user capabilities on DOS 2.0 (and above)
- Indexed and direct files
- Access to host system text (serial) files
- Record and file locking
- Trigonometric, logarithmic and exponential functions
- Menu driven resource configurator
- Multidimensional arrays and vectors
- Binary and string logical functions
- Extended position function
- Public programming
- Supports ghost tasks
- Fully formatted output
- Up to 255 files per program

All of this for only **$595.00.** CALL TODAY!

SOUTHWEST DATA SYSTEMS, INC. 3017 San Fernando Blvd., Burbank, CA 91504 • (818) 841-1610

CIRCLE 46 ON READER SERVICE CARD

**SPARRY BASIC-B COMPILER**

NEW 2.0 Release

1. Floating Point Math
2. Use all 640K of Memory
3. Multiple Data Segments
4. Multiple Code Segments
5. Internal ISAM File support
6. 4 Virtual Screens (Big Windows)
7. Easy Assembly Language Interface
8. Direct System Interrupt Calls
9. A Compatible BASIC Compiler

Req. PCDOS 2.00 + with 128K

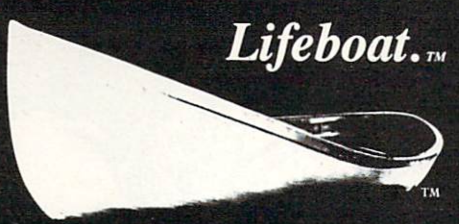
**Sparry Software Labs**

P.O. BOX 632
MILFORD, MA 01757
617-473-5435

( ) Compiler $159
( ) Demo Disk $15

PCDOS is a Trademark of
International Business Machine Corp.

CIRCLE 66 ON READER SERVICE CARD



*C is the language.*
*Lifeboat™ is the source.*

# Productivity Tools from the Leading Publisher of C Programs.™

## The Lattice® C Compiler

The cornerstone of a program is its compiler; it can make the difference between a good program and a great one. The Lattice C compiler features:

- Full compatibility with Kernighan and Ritchie's standards
- Four memory model options for control and versatility
- Automatic sensing and use of the 8087 math chip
- Choose from the widest selection of add-on options
- Renowned for speed and code quality
- Superior quality documentation

"Lattice C produces remarkable code. . . the documentation sets such a high standard that others don't even come close. . . in the top category for its quick compilation and execution time and consistent reliability."

*Byte* Magazine

Lattice Library source code also available.

## Language Utilities

**Pfix 86/Pfix 86 Plus** — dynamic and symbolic debuggers respectively, these provide multiple-window debugging with breakpointing capability.
**Plink 86** — a two-pass overlay linkage editor that helps solve memory problems.
**Text Management Utilities** — includes GREP (searches files for patterns), DIFF (differential text file comparator), and more.
**LMK (UNIX "make")** — automates the construction of large multi-module products.
**Curses** — lets you write programs with full screen output transportable among all UNIX, XENIX and PC-DOS systems without changing your source code.
**BASTOC** — translates MBASIC or CBASIC source code directly to Lattice C source code.
**C Cross Reference Generator** — examines your C source modules and produces a listing of each symbol and where it is referenced.

## Editors

**Pmate** — a customizable full screen text editor featuring its own powerful macro command language.
**ES/P for C** — C program entry with automatic syntax checking and formatting.
**VEDIT** — an easy-to-use word processor for use with V-PRINT.
**V-PRINT** — a print formatting companion for VEDIT.
**CVUE** — a full-screen editor that offers an easy way to use command structure.
**EMACS** — a full screen multi window text editor.
**Fast/C** — speeds up the cycle of edit-compile-debug-edit-recompile.

## Graphics and Screen Design

**HALO** — one of the industry's standard graphics development packages. Over 150 graphics commands including line, arc, box, circle and ellipse primitives. The **10 Fontpack** is also available.
**Panel** — a screen formatter and data entry aid.
**Lattice Window** — a library of subroutines allowing design of windows.

## Functions

**C-Food Smorgasbord** — a tasty selection of utility functions for Lattice C programmers; includes a binary coded decimal arithmetic package, level 0 I/O functions, a Terminal Independence Package, and more.
**Float-87** — supports the 8087 math chip to boost the speed of floating-point calculations.
**The Greenleaf Functions** — a comprehensive library of over 200 routines.
**The Greenleaf Comm Library** — an easy-to-use asynchronous communications library.
**C Power Packs** — sets of functions useful for a wide variety of applications.
**BASIC C** — This library is a simple bridge from IBM BASIC to C.

## Database Record Managers

**Phact** — a database record manager library of C language functions, used in the creation and manipulation of large and small databases.
**Btrieve** — a sophisticated file management system designed for developing applications under PC-DOS. Data can be instantly retrieved by key value.
**FABS** — a Fast Access Btree Structure function library designed for rapid, keyed access to data files using multipath structures.
**Autosort** — a fast sort/merge utility.
**Lattice dB-C ISAM** — a library of C functions that enables you to create and access dBase format database files.

## Cross-Compilers

For programmers active in both micro and mini environments we provide advanced cross-compilers which product Intel 8086 object modules. All were developed to be as functional — and reliable — as the native compilers. They are available for the following systems:

**VAX/VMS, VAX/UNIX, 68K/UNIX-S, 68K/UNIX-L**

Also, we have available:
**Z80 Cross-Compiler for MS- and PC-DOS** — produces Z80 object modules in the Microsoft relocatable format.

## New Products

**Run/C** — finally, a C interpreter for all levels of C Programmers.
**C Sprite** — a symbolic debugger with breakpoint capability.

***Call LIFEBOAT: 1-800-847-7078. In NY, 1-212-860-0300.***

CIRCLE 85 ON READER SERVICE CARD

*YES! Please rush me the latest FREE Lifeboat™ catalog of C products.*

*Name*________________ *Title*________________

*Company Name*________________ *Business Phone*________________

*Address*________________

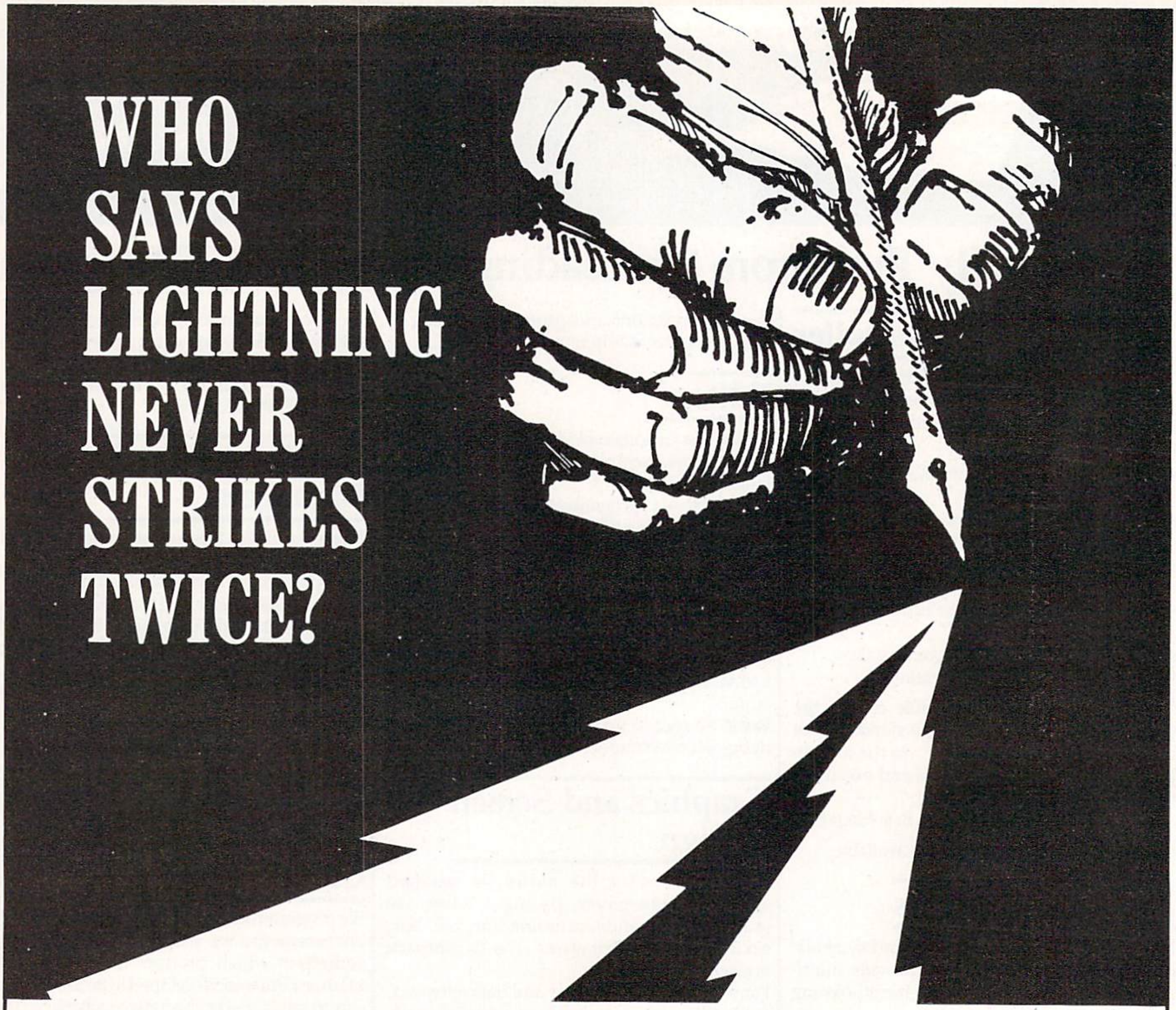
*Please check one of the following categories:*
☐ *Dealer/Distributor* ☐ *End User* ☐ *Other*________

*Return Coupon to: Lifeboat™ Associates*
*1651 Third Avenue, New York, NY 10128*

CL

© 1985 Lifeboat Associates

C Programming Tools
Lifeboat
Associates


WHO SAYS LIGHTNING NEVER STRIKES TWICE?

Whitesmiths sets yet another precedent
with the *first* in a series of new portable standard C Compilers:
**C for the 8086 family**

Features:

- Includes Pascal which conforms to *full* ISO (level 1) standard, plus popular extensions and long identifiers
- C now has struct assignment, enumerations, plus other popular features and long identifiers
- Supports all memory models from small to large, *plus mixed pointer sizes and segment overrides*
- Source level portable debugger included
- Generates assembler listings with intermixed source code
- Multi-segment linker with direct *or* sequential libraries, plus librarian, assembler, and other object tools included
- Source code of system interface library included
- Library use fees included in purchase price

Whitesmiths, Ltd. • 97 Lowell Road • Concord, MA 01742 • 617-369-8499 • Telex 750246

CIRCLE 39 ON READER SERVICE CARD

# EXOTIC LANGUAGE OF THE MONTH CLUB

## Clascal—An object-oriented Pascal

**By Tim Endres**

In 1983 Apple Computer set into motion a new ideal in the personal computer industry. With the introduction of the Lisa, Apple gave the computer public its first inoculation of user-friendly, fully integrated software.

No other personal computer has quite matched the innovation of the Lisa. Yet very little is known about the heart of this computer. Besides some of the most impressive hardware specifications for a computer in its price category, the Lisa has some of the most sophisticated software ever packed into a personal computer.

Included in this software is a very fast and powerful graphics driver (QuickDraw), a standard operating system, a hardware interface, and a development environment called the Workshop—with Pascal, Assembler, Linker, COBOL, BASIC, and a command list processor.

Apple used these standard building blocks, along with several custom utility libraries designed to drive the windows and folders of the desktop, to create the Office System environment. The Office System is a collection of integrated software aimed at office and business automation.

To top the package off, Apple spent a great deal of effort to develop a system that would allow future programs written for the Office System to be fully integrated. This system, called the Toolkit, was built on an object-oriented language named Clascal.

Clascal was an extension built into Apple's Pascal that gave it an object-oriented syntax. Clascal and the Toolkit provided me, an independent developer, with the resources to develop an application for the Office System in approximately half the development time I would have expected for any other PC. This application was fully integrated into the Office System and provided a user-friendly interface consistent with all other Office System applications.

To realize the significance of the Toolkit, one must first appreciate Clascal. Clascal is an extension of Pascal that makes it object oriented, which means that the syntax of the program is based on the idea of objects instead of procedures and object parameters instead of variables.

With regular Pascal, you program using procedures, functions, and variables. With Clascal, you program using objects, which have methods (procedures and functions) and data fields (parameters).

To illustrate the difference between Pascal and Clascal, consider the code written to display and control the window in which my application executed (Figure 1). The window had three different views in which user interaction occurred: the status view on top, calendar view bottom left, and appointment view bottom right.

Let's consider writing the code that would be responsible for merely drawing the window. In Pascal, I would call the procedure that draws the window:

```
Draw_Window;
```

The *Draw_Window* procedure would call the procedures responsible for drawing each of the views and highlighting selections:

```
PROCEDURE Draw_Window;
    BEGIN
        Draw_Status_View;
        Draw_Appointment_View;
        Draw_Calendar_View;
        Hilite_Selections;
        END;
```

In Clascal, the window is an object. Objects are referenced with symbolic variables the same way variables are referenced in Pascal. To draw the object, you must invoke its method, called *Draw*:

```
myWindow.Draw;
```

Here *myWindow* is a symbolic reference to the window object. *MyWindow*'s *Draw* method would first call each view's *Draw* method, then invoke its own highlighting method. In Listing 1, notice how the data field *views* is a reference to an object that is a list of objects. Each object in the list is a view of the window. The method *Each* invokes the method in parenthesis for each object in the list. Lists are important classes in the Toolkit.

The keyword *SELF* is essential to Clascal. When any method in Clascal uses this keyword, it is referencing the object that was asked to perform the method (*myWindow* in the preceding example). To

```
PROCEDURE TStaWindow.Draw;    {TStaWindow is myWindow's class name}
   BEGIN
      SELF.views.Each(Draw);  {Causes each object in list "views" }
                              {to invoke its DRAW method         }
      SELF.HiliteSel;         {Invokes the window's own HiliteSel
                              {method                            }
         END;
```

Listing 1.

understand this a little more, we must discuss classes, the foundation of Clascal.

Objects are the functional building blocks for Clascal. Every tangible piece of the program is represented by an object, and every tangible action in the program is the method of an object or a function or procedure called by a method.

Thus, when the software developer writes code, he or she is creating objects that represent the different pieces of the application (for example, *Window, View, Appointment, Calendar, Day, Clock, Folder*) and causing these objects to act upon each other by invoking their methods.

Classes are the conceptual building blocks of Clascal. Every object created in a program is defined to be in a particular class. Classes are similar in syntax to types in Pascal but function considerably differently. They create one of the most powerful aspects of Clascal.

Classes describe the types of objects used in a program in terms of the parameters of each object and the methods that each one can perform. More importantly, each class is a subclass of some other class. This is a required syntax and gives the language a hierarchical structuring.

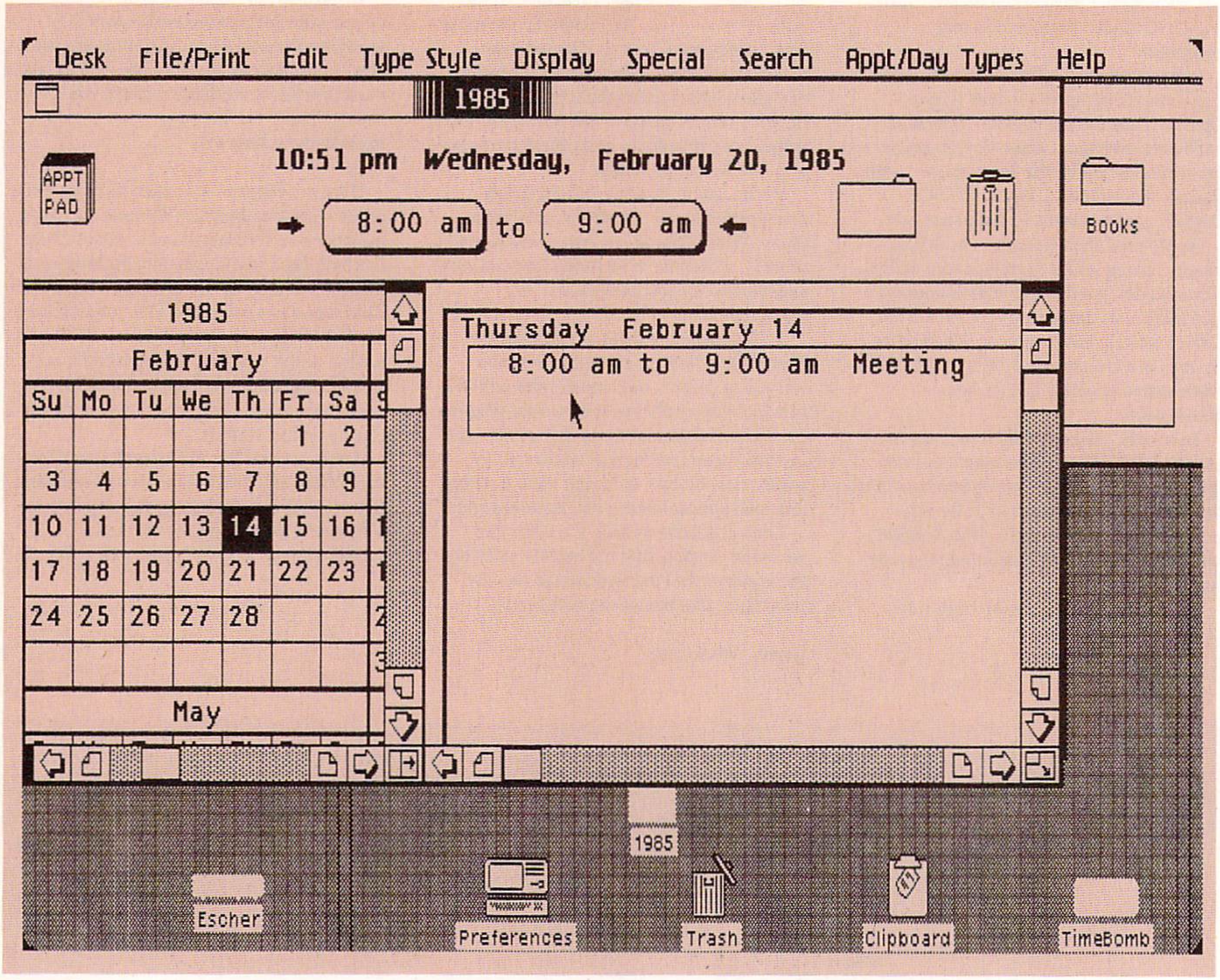
This hierarchy is one of the most powerful aspects of Clascal. It accounts for a very critical quality called extendibility. Extendibility is the ability to take a functioning block of code and extend its capabilities through the mechanism of subclassing.

For instance, again in Listing 1, I used the object *myWindow* to control the window of my application. This object belongs to the class *TStaWindow*, a subclass of *TWindow*, which is a class provided in the Toolkit. Any object in the class *TWindow* can perform the following methods:

- Create a new object of the class *TWindow*
- Delete itself
- Clone or duplicate itself
- Draw itself
- Perform commands and handle mouse and keyboard events
- Open, close, suspend, resize, and refresh itself.

Because of inheritance, every object in the class *TStaWindow* can perform any of the preceding methods. The objects of the class *TStaWindow* also inherit the parameters defined for objects of the class *TWindow*. Thus, by merely coding the line that states *TStaWindow* is a subclass of *TWindow*, I can create my own windows that do everything Apple has coded for objects of the class *TWindow*.

Better still, I can add to the parameters and methods that are defined for *TWindow* objects to make my windows more functional. Plus, I may redefine the methods that are already defined for *TWindow* objects to do something different. For example, Apple's *TWindow* objects, when



10:51 pm Wednesday, February 20, 1985

8:00 am to 9:00 am

1985

February

| Su | Mo | Tu | We | Th | Fr | Sa |
|---|---|---|---|---|---|---|
|  |  |  |  |  | 1 | 2 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 | 28 |  |  |
|  |  |  |  |  |  |  |

May

Thursday February 14

8:00 am to 9:00 am Meeting

Figure 1.

activated, would do some default highlighting of selected objects. This was fine except that the default highlighting logic caused a glitch in my program's display in certain cases. To fix this problem, I simply redefined the method that activated the window to use a different highlighting logic. I did this by defining a method for the class *TStaWindow* with the same name as the method defined for *TWindow*.

Why did this override the activation method performed by *TWindow*? This brings us back to the keyword *SELF*. Whenever a Clascal method uses the keyword *SELF*, it is referencing the object that was asked to perform the method.

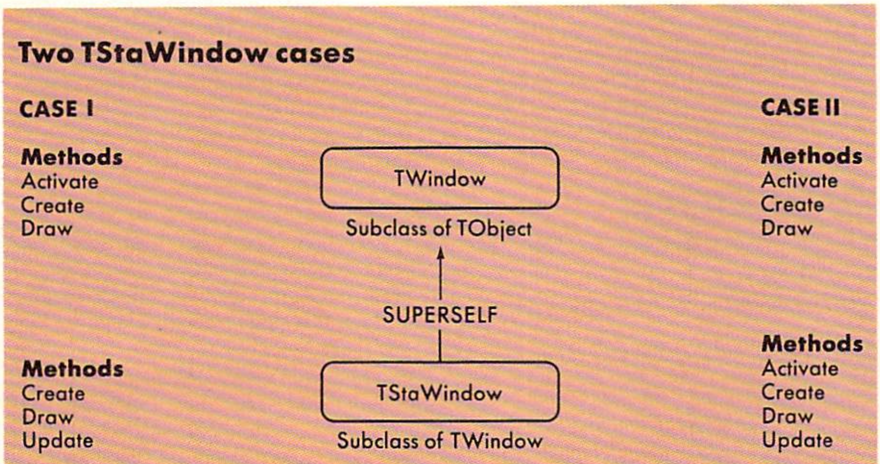
Notice the number of occurrences of the word "itself" in the list of the previous methods. Consider, from the example method *TStaWindow.Draw*, the line *SELF.HiliteSel*. This line states that the window object should perform its method called *HiliteSel*. When the keyword *SELF* is encountered, a search beginning with the object performing the method is conducted for the method definition. If the method is not defined for the object's class, the search continues up through the superclasses of the object's class, until the definition of the method is found.

If I define my own method called *HiliteSel* for the *TStaWindow* class, and a *TStaWindow* object is asked to perform its *Draw* method, then that method in turn will invoke its own *HiliteSel* method. If I had not defined this method for *TStaWindow* objects, then the *Draw* method would invoke the *HiliteSel* method defined for *TWindow* objects. This situation is illustrated in Figure 2.

Just as methods are inherited, so are data fields. The methods defined for *TStaWindow* objects can reference the same data fields defined for *TWindow* objects. The only difference is that you do not override data fields. Inheritance is collective. Classes inherit the methods and parameters of their superclass, which in turn inherit the methods and parameters of their superclasses.

The Toolkit is a library of software provided to support Office System application development. It accomplishes this through the mechanism of extendibility provided by Clascal.

The Toolkit is a collection of class definitions that together perform all of the basic Office System functions. It defines windows, views, panels, scroll bars, documents, document managers, processes,



| | |
|---|---|
| **TStaWindow object:** | **SELF.Activate;** |
| Search begins in the TStaWindow class. No method is defined for this class, so the search goes up to the TWindow class and executes the method TWindow.Activate. (Inherent) | Search begins in the TStaWindow class. Since this class defines the method Activate, the method TStaWindow.Activate is executed. (Extension) |
| **TWindow object:** | **SELF.Activate;** |
| Search begins in the TWindow class. Since this class defines the method Activate, the method TWindow.Activate is executed. | Search begins in the TWindow class. Since this class defines the method Activate, the method TWindow.Activate is executed. |

Figure 2.

C

C LIBRARIES
C WINDOWS

Best You Can Get!
325 Fully Tested Functions
SIX C LIBRARIES

FUNCTIONS YOU DON'T HAVE BUT NEED!
All Source Code. No royalties.

| | | |
|---|---|---|
| 57 | screen handling/graphic functions | $49.95 |
| 50 | cursor/keyboard/data input functions | $39.95 |
| 85 | superior string functions | $59.95 |
| 31 | system status & control functions | $29.95 |
| 72 | utility/DOS/BIOS/time/date functions | $49.95 |
| 42 | printer control functions | $29.95 |

C-TO-FORTRAN/FORTRAN-TO-C
RICHLY COMMENTED
Easy to Learn/Easy to Modify
Execute other Programs Internally

No Matter What Else You Have, Get These!
ANY 3 LIBRARIES $69.95
ALL 6 LIBRARIES $99.95

50 MOST NEEDED FUNCTIONS
$49.95
3270 FUNCTION PACKAGE $69.95

C WINDOWS
PROFESSIONAL WINDOW MANAGEMENT
Overlays,Borders,Popup Menus,Help Windows,
Status-Line,Color Highlighting And More!!!
C Windows: Complete Source Code.....$89.95

THE PROFILER
by DWB ASSOCIATES
The Cadillac of profilers..$125.00

COMBINATION OFFER
C WINDOWS PLUS 6 LIBRARIES
For $149.95
SIX LIBRARIES & THE PROFILER
Both For $179.95
C WINDOWS & 6 C LIBRARIES
& THE PROFILER
A $315 Value All For $219.95

Entelekon
SOFTWARE SYSTEMS
ENTELEKON 12118 KIMBERLEY
HOUSTON, TX. 77024 (713)-468-4412

CIRCLE 50 ON READER SERVICE CARD

selections, clipboards, dialog boxes, text and much more. These definitions provide the software developer with an application skeleton, called the generic application.

By coding no more than 100 lines, I can develop an application that will display a window with scroll bars, a view with my name drawn in it, and menus. It would allow moving and sizing of the window, picking of commands from menus, scrolling, view splitting, and document creation, saving, and deleting. Then, by overriding defined methods and creating my own classes, my application slowly develops into its own unique definition.

Since Toolkit classes were provided for cutting, pasting, printing, mouse handling, and text, my application was fully integrated with all other Office System applications. The application also utilized similar mechanisms for the user interface.

The amount of coding I was saved by not having to develop Toolkit classes is quite significant. The source code of the Toolkit contains approximately four times the number of lines that my application contains. Since development took me nine months, I figure the Toolkit saved me at least two years. It also provided the less exciting, low-level code and allowed me to develop at a higher, more creative level.

Clascal has brought object-oriented languages into a new arena. For the first time, general PC programmers can develop software with an object-oriented language. For the first time, they might learn what SmallTalk is. Apple has just released a language called Object Pascal for the Macintosh and created MAC App, the equivalent of the Toolkit. Cross your fingers in the hopes that more businesses will see that object-oriented languages are the next language in the hierarchy.

This language also provides an inherent structuring. The different conceptual objects of an application must belong to classes. Each class has a set of methods, which perform functions unique to the class. Each class has a set of parameters. These classes must be hierarchical and inherit methods and parameters from parent classes.

Clascal, being an object-oriented language, reduces development time and increases integration, due to extendibility. Extendibility also provides a convenient mechanism to provide generic expert systems that users can customize through subclassing.

Clascal is truly an unsung hero. Of all the fanfare the Lisa received, only a handful of articles even mentioned the Toolkit, and fewer yet talked about Clascal. If more exposure is given to this language, perhaps this situation will change.

*Tim Endres is responsible for advanced planning and technology transfer for GM/EDS at Buick-Oldsmobile-Cadillac in Lansing, Mich. He has a B.S. in electrical engineering from General Motors Institute and is a licensed developer for Apple's Lisa and Macintosh.*

C Programmers Quit Working So Hard!

C
THE GREENLEAF FUNCTIONS™
GREENLEAF SOFTWARE

**THE GREENLEAF FUNCTIONS™**

**The GREENLEAF FUNCTIONS GENERAL LIBRARY** has over 200 functions in C and assembler. Strength in DOS, video, string, printer, async, and systems interface. All DOS 1 and 2 functions are in assembler for speed. All video capabilities of PC supported. All printer functions. 65 string functions. Extensive time and date. Directory searches. Polled mode async. (If you want interrupt driven, ask us about the **Greenleaf Comm Library.**) Function key support. Diagnostics. Rainbow Color Text series. Much, much more. **The Greenleaf Functions.** Simply the finest C library (and the most extensive). All ready for you.

**THE GREENLEAF FUNCTIONS™**

The Library of C Functions that probably has just what you need . . . **TODAY!**

- already has what you're working to re-invent
- already has over 200 functions for the IBM PC, XT, AT, and compatibles
- already complete . . . already tested . . . on the shelf
- already has demo programs and source code
- already compatible with all popular compilers
- already supports all memory models, DOS 1.1, 2.0, 2.1
- already optimized (parts in assembler) for speed and density
- already in use by thousands of customers worldwide
- already available from stock (your dealer probably has it)
- It's called the **GREENLEAF FUNCTIONS.**

**The Library of C Functions is Waiting for You**

**Specify compiler** when ordering. Add $7.00 for UPS second-day air (or $5.00 for ground). Texas residents add sales tax. Mastercard, VISA, check or P.O. In stock, shipped same day.

For Information: 214-446-8641

| | |
|---|---|
| General Libraries | $185 |
| Comm Library | $185 |
| CI C86 Compiler | $349 |
| Lattice C | $395 |
| Mark Williams | $475 |

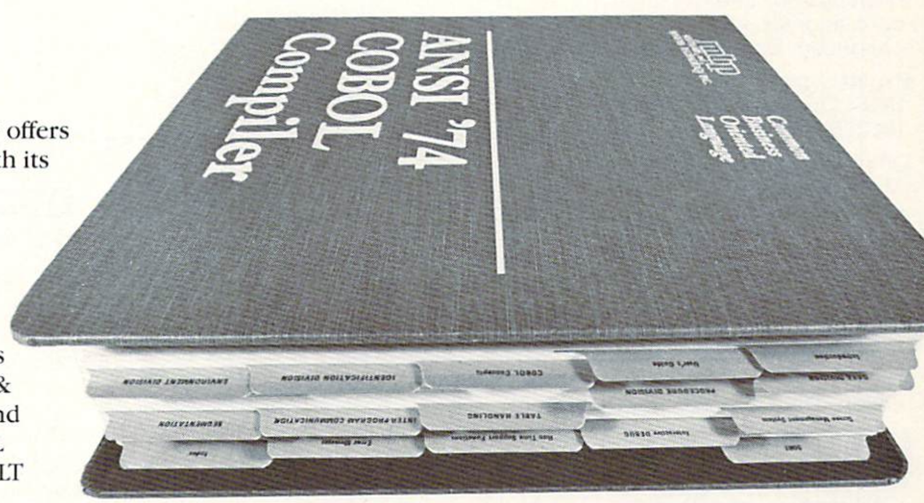
Prices are subject to change without notice.

GREENLEAF SOFTWARE©
2101 HICKORY DR.
CARROLLTON, TX 75006

CIRCLE 44 ON READER SERVICE CARD

mbp COBOL
for your IBM/PC

# The new standard for convenience.



Now, the mbp COBOL Compiler offers unrivaled convenience to go with its unmatched performance.

**Here are the convenience features you've wished for:**

1) an enhanced Screen Management System with program-controlled video attributes and color; 2) support for PATH & sub-directories; 3) DOS command execution from within a COBOL program; 4) 'permanent' DEFAULT modification.

The new mbp Compiler has them all! And they're exclusives: you get them *only* with mbp.

**Plus, it's 4 times faster.**

Because the mbp COBOL Compiler generates native machine language object code, it executes programs *at least* 4 times faster (see chart). Now, we've made that performance even more convenient to use.

**GIBSON MIX Benchmark Results**
Calculated S-Profile
*(Representative COBOL statement mix)*
**Execution time ratio**

| **mbp* COBOL** | Level II* COBOL | R-M* COBOL | Microsoft* COBOL |
|---|---|---|---|
| **1.00** | 4.08 | 5.98 | 6.18 |

*128K system with hard disk required. IBM PC & AT are IBM TMs; Netware is a Novell, Inc TM; Level II is a Micro Focus TM; R-M is a Ryan-McFarland TM; Microsoft is a Microsoft TM.

**The complete COBOL.**

An Interactive Symbolic Debug Package included standard; Multi-keyed ISAM structure; SORT & CHAIN; GSA certification to ANSI '74 Level II; IBM/PC-AT and TI Professional compatibility; with mbp, you get it all. Optional: Novell NetWare interface.

**mbp COBOL: the choice of professionals.**

It's no surprise more and more companies like Bechtel, Bank of America, Chase, Citicorp, Connecticut Mutual, Hughes Aircraft, McDonnell-Douglass, and Price-Waterhouse choose mbp COBOL.

*Make it your choice, too.* Just send the coupon, or call, for complete information. Today.

## mbp COBOL. $1000

*Please send complete mbp COBOL information to:*

NAME ____________________

COMPANY ____________________

ADDRESS ____________________

CITY/STATE/ZIP ____________________

PHONE ____________________

**mbp Software & Systems Technology, Inc.**
7700 Edgewater Drive, Suite 360
Oakland, CA 94621
Phone 415/632-1555

mbp®

CIRCLE 53 ON READER SERVICE CARD

# THE PROGRAMMER'S SHOP™

helps compare, evaluate, find products. Straight answers for serious programmers.

**SERVICES**

- **Programmer's Referral List**
- **Compare Products**
- **Help find a Publisher**
- **Evaluation Literature free**
- **BULLETIN BOARD - 7 PM to 7 AM 617-826-4086**
- **Dealer's Inquire**
- **Newsletter**
- **Rush Order**
- **Over 700 products**

## Free Literature - Compare Products

Evaluate products Compare competitors. Learn about new alternatives. One *free call* brings information on just about any programming need. Ask for any ***"Packet"*** or "Addon Packet": ☐ ADA, Modula ☐ "AI" ☐ BASIC ☐ "C" ☐ COBOL ☐ Editors ☐ FORTH ☐ FORTRAN ☐ PASCAL ☐ UNIX/PC or ☐ Debuggers, Linkers, etc.

## RECENT DISCOVERIES

FASTER C - Lattice users eliminate Link step. Normal 27 seconds, Faster C in 13 secs. MSDOS $95

## ARTIFICIAL INTELLIGENCE

EXSYS - Expert System building tool. Full RAM, Probability, Why, Intriguing, serious. PCDOS $275

GC LISP - "COMMON LISP", Help, tutorial, co-routines, compiled functions, thorough. PCDOS $455

IQ LISP - MACLISP & INTERLISP. Full RAM. Liked. PCDOS $155

TLC LISP - "LISP-machine"-like, all RAM, classes, turtle graphics 8087. CP/M-86, MSDOS $235

INSIGHT 1 - Expert Sys. Dev't, decent PCDOS $95

PROLOG-86 - Learn fast, Standard, tutorials, samples of Natural Language, Exp. Sys. MSDOS $125

Expert System front-ends for PROLOG: APES ($275), ES/P ($1895)

Other solid alternatives include: MuLISP-86 ($189), WALTZ LISP for CPM ($159), MicroPROLOG ($275)

## EDITORS FOR PROGRAMMING

BRIEF Programmer's Editor - undo, windows, reconfigurable, macro programs, powerful. PCDOS $195

VEDIT - well liked, macros, buffers, CPM-80-86, MSDOS, PCDOS $119

## MACINTOSH

We evaluate, carry every available programmers product. Ask.

## C LANGUAGE

INSTANT C - Interactive development - Edit, Source Debug, run. Edit to Run - 3 Secs. MSDOS $495

"INTRODUCING C" - Interactive C to learn fast. 500 page tutorial, examples, graphics. PCDOS $95

MEGAMAX C - native Macintosh has fast compile, tight code, K&R, toolkit, .OBJ, DisASM MAC $275

Audio-based C tutorials. Overview $95. Full $295

## C LIBRARIES

COMMUNICATIONS by Greenleaf ($159) or Software horizons ($139) includes Modem7, interrupts, etc. Source. Ask for Greenleaf demo.

C SHARP Realtime Toolkit - well supported, thorough, portable, objects, state sys. Source MANY $600

APPLICATION TOOLKIT by Shaw - Complete: ISAM, Screen, Overlay mgnt, report gen, Strings, String math. Source. CPM, MSDOS $495

ROMPack - special $Main .EXE editor, source, tech support, 8086. $185

## DEBUGGERS

PERISCOPE DEBUGGER - load after "bombs", symbolic, "Reset box", 2 Screen, own 16K. PCDOS $279

SOURCE PROBE by Atron for Lattice, MS C, Pascal. Windows single step, 2 screen, log file. $395

## FORTRAN LANGUAGE

MacFORTRAN - full '77, '66 option, toolbox, debugger, 128K or 512K, ASM-out option MAC $375

DR/Fortran-77 - full ANSI 77, 8087, overlay, full RAM, big arrays, complex NUMS., CPM86, MSDOS $249

Ask about Microsoft, Supersoft, others.

## OTHER LANGUAGES

ASSEMBLER - ask about FASM-86 ($95), ED/ASM ($100) - both are fast, compatible, or MASM ($125), improvements.

BetterBASIC all RAM, modules, structure. BASICA - like $185

HS/FORTH - '79 & '83 Standards, full RAM, ASM, BIOS, interrupts, graph, multi-task, optimizer MSDOS $250

MBP COBOL has screen control, strong doc, '74 interm., fast. MSDOS $680

## SUPPORT PRODUCTS

BASIC DEVELOPMENT SYSTEM - (BDS) for BASICA; Adds Renum, crossref, compress. PCDOS $115

PLINK-86 for Overlays, most lang., segment control. MSDOS $325

ProYAM Communications Package - All a programmer'd want. TTY, VT 100, 3101, MODEM7, BBS. Remote, macros, windows MSDOS $139

CODESMITH - visual, interactive debugger. Symbolize, modify code $129

## "C" LANGUAGE

| | OUR PRICE |
|---|---|
| MSDOS: C86-8087, reliable | call |
| Instant C - Inter., fast, full | 495 |
| Lattice 2.1 - improved | call |
| Microsoft C 2.x | 279 |
| Williams, debugger, fast | call |
| C Systems & debugger | 175 |
| CPM80: EcoPlus C - faster, SLR | 275 |
| BDS C - solid value | 125 |
| MACINTOSH: Softworks | 365 |
| Megamax-object, full | 275 |
| Consulair's MAC C | 275 |

Compare, evaluate, consider other Cs

## BASIC

| | RUNS ON | |
|---|---|---|
| Active Trace-debug | 86/80 | 75 |
| BASCOM-86 - MicroSoft | 8086 | 279 |
| BASIC Dev't System | PCDOS | 115 |
| BetterBASIC - 640K | PCDOS | 185 |
| CB-86 - DRI | CPM86 | 419 |
| Prof. BASIC Compiler | PCDOS | 89 |
| Databurst - screens | MSDOS | 215 |
| SCREEN SCULPTOR | PCDOS | 115 |

Ask about ISAM, other addons for BASIC

## SERVICE

**ALL PRODUCTS** - We carry 700 products for MSDOS, CP/M 86, CP/M 80, MacIntosh and key products for other micros.

## EDITORS Programming

| | RUNS ON | OUR PRICE |
|---|---|---|
| BRIEF - Intuitive, flexible | PCDOS | 195 |
| C Screen with source | 86/80 | 75 |
| Epsilon - like EMACS | PCDOS | 195 |
| FINAL WORD-for manuals | 86/80 | 215 |
| MINCE-like EMACS | PC/80 | 149 |
| PMATE-powerful | 8086 | 185 |
| VEDIT-full, liked | 86/80 | 119 |

## UNIX™ PC

| | | |
|---|---|---|
| COHERENT - for "C" users | PClike | 475 |
| COHERENT-NCI-Realtime | PClike | call |
| **XENIX** - plus C to MSDOS | PC | 1275 |

Ask about run-times, applications, DOS compatibility, other alternatives. UNIX is a trademark of Bell Labs

## LANGUAGE LIBRARIES

| | | |
|---|---|---|
| GRAPHICS: GraphiC-source in C | MSDOS | 219 |
| GRAPHMATIC-3D, FTN, PAS | PCDOS | 125 |
| HALO-fast, full-all lang. | PCDOS | 139 |
| FILE MGNT: BTrieve-all lang. | MSDOS | 215 |
| CIndex + -source, no royal. | 86/80 | 369 |
| CTree-source, no royal. | ALL | 369 |
| dBC ISAM by Lattice | 8086 | 229 |
| dB VISTA-"Network" Structure | MSDOS | 465 |
| PHACT-up under UNIX, addons | MSDOS | 225 |
| OTHER: CUtil by Essential | MSDOS | 129 |
| Greenleaf - 200 + | MSDOS | 159 |
| CSharp - **Real-Time** | MSDOS | 600 |
| PORTABLE C to PC, Mac, II | Many | 125 |
| SOFT Horizons - Blocks I | PCDOS | 139 |
| SCREEN: CURSES by Lattice | PCDOS | 125 |
| CView - input, validate | PCDOS | 195 |
| MetaWINDOW - icons, clip | PCDOS | 139 |
| PANEL - many lang. term | MSDOS | 249 |
| ProScreen - windows, source | PCDOS | 415 |
| Windows for C | MSDOS | 175 |

## FORTRAN

| | RUNS ON | OUR PRICE |
|---|---|---|
| MS FORTRAN-86 - Impr. | MSDOS | $ 239 |
| DR Fortran-86 - full '77' | 8086 | 249 |
| PolyFORTRAN-XREF, Xtract | PCDOS | 165 |

## OTHER PRODUCTS

| | | |
|---|---|---|
| Assembler & Tools - DRI | 8086 | 159 |
| Atron Debugger for Lattice | PCDOS | 395 |
| cEnglish - dBase to C | MSDOS | 750 |
| C Helper: DIFF, xref, more | 86/80 | 135 |
| CODESMITH-86 - debug | PCDOS | 1 9 |
| MacASM-full, fast, tools | MAC | 115 |
| MBP Cobol-86 - fast | 8086 | 680 |
| Modula 2 for MAC, | PCDOS | 90 |
| Micro: SubMATH-FORTRAN full | 86/80 | 250 |
| Microsoft MASM-86 | MSDOS | 125 |
| MSD Debugger | PCDOS | 119 |
| Multilink - Multitasking | PCDOS | 265 |
| PC/FORTH + -well liked | MSDOS | 219 |
| PFIX-86 Debugger | MSDOS | 169 |
| PL/1-86 | 8086 | 495 |
| Polylibrarian - thorough | MSDOS | 95 |
| PolyMAKE | PCDOS | 95 |
| PROFILER by DWB - flexible | MSDOS | 109 |
| Prolog-86-Learn, Experiment | MSDOS | 125 |
| SLK/F - Copy Protection | PCDOS | 145 |
| SYMD debugger-symbols | PCDOS | 119 |
| TRACE86 debugger ASM | MSDOS | 115 |

Note: All prices subject to change without notice. Mention this ad. Some prices are specials.
Ask about COD and **POs**. All formats available.

**Call for a catalog, literature, and *solid value***

# 800-421-8006

**THE PROGRAMMER'S SHOP™**
128-L Rockland Street, Hanover, MA 02339
Visa Mass: 800-442-8070 or 617-826-7531 MasterCard 8517

CIRCLE 40 ON READER SERVICE CARD

# PRODUCT BINGO

**By Doug Millison**

*Each month Product Bingo features the latest in new software and hardware products of interest to COMPUTER LANGUAGE readers. Product Bingo items are based on information received from the manufacturer and are not meant to be product evaluations, reviews or endorsements. To find out more about a particular product simply circle the appropriate number on the Reader Service card—you'll receive information directly from the manufacturer.*

*Note to manufacturers: Send new product information to Doug Millison, Product Bingo, COMPUTER LANGUAGE, 131 Townsend St., San Francisco, Calif. 94107.*

## Compiler kit for UNIX

**Unipress Software Inc.** has introduced the **Amsterdam Compiler Kit** for the UNIX operating system. The kit compiles C and Pascal on UNIX 68000, 8086, VAX, and PDP-11 and produces code for any of these machines.

Cross assemblers are provided for 8080, Z8000, 8086, 6800, 6809, 68000, 6502, and PDP-11. Cross interpreters are also included for testing.

Developed at Vrije Univ. of Amsterdam, the Netherlands, the Amsterdam Compiler Kit is priced at $9,950 ($995 for educational institutions), including all sources and documentation. Selected binaries are priced at $4,500.

UniPress Software Inc., 2025 Lincoln Highway, Ste. 312, Edison, N.J. 08817, (201) 895-8000.

**CIRCLE 101 ON READER SERVICE CARD**

## Utilities for PC BASIC

From **Timeshare Associates Inc.** comes a series of utilities designed to aid the programmer working in PC BASIC.

**AUTOCPL** is a precompiler system allowing programs to be run in the BASIC interpreter, then compiled and linked quickly with no manual code changes. **AUTOTRACE** will trace an interpreted or compiled program in single-step or continuous mode.

**AUTOCOMPARE** prints or saves to disk all differences between two files and can convert word processor to ASCII files. **AUTOREMARK** removes REMarks from programs, lists programs with REMarks highlighted on the right, and restores REMarks to a program.

Each utility comes with RAM disk and spooler and retails for $49.95. The price is reduced by one-third when the utilities are purchased in combinations.

Timeshare Associates Inc., 1020 Robinson, Overland Park, Kan. 66212, (913) 642-7564.

**CIRCLE 102 ON READER SERVICE CARD**

## CBASIC programmers do it faster

**Minnow Bear Computers** promises CBASIC programmers enhanced productivity and faster-running programs with **CBC Tools**.

A large set of functions that can be called from the Digital Research CBASIC compiler, CBC Tools is available for use under CP/M 80 and CP/M 86. Most functions are written in assembly language. A full-featured program tracer and program profiler are included.

Available in popular 8-in. and 5¼-in. disk formats, CBC Tools is priced at $105, freight included.

Minnow Bear Computers, 604 E. Clark St. #5, Champaign, Ill. 61820, (217) 398-6883.

**CIRCLE 105 ON READER SERVICE CARD**

## BASIC programmers tap UNIX, C

**UX-BASIC+,** an advanced version of a multiuser, multitasking business language first used on computers supplied to NASA for the space shuttle program, is now available from **UX Software Inc.**

Providing access to C subroutine libraries, UNIX system calls, C-ISAM and UNIX graphics, UX-BASIC+ also interacts directly with UNIX facilities and may be used as a shell providing friendly user-to-UNIX interaction.

UX-BASIC+ is said to allow programmers familiar with BASIC to develop C-driven vertical market applications. Run-time systems for a variety of computers, ranging from the IBM PC/XT to the VAX 11/750, retail for $150 to $900. Development systems retail from $650 to $3,900.

UX Software Inc., 10 St. Mary St., Toronto, Ont., Canada M4Y 1P9, (416) 964-6909.

**CIRCLE 103 ON READER SERVICE CARD**

## BASIC windows and more

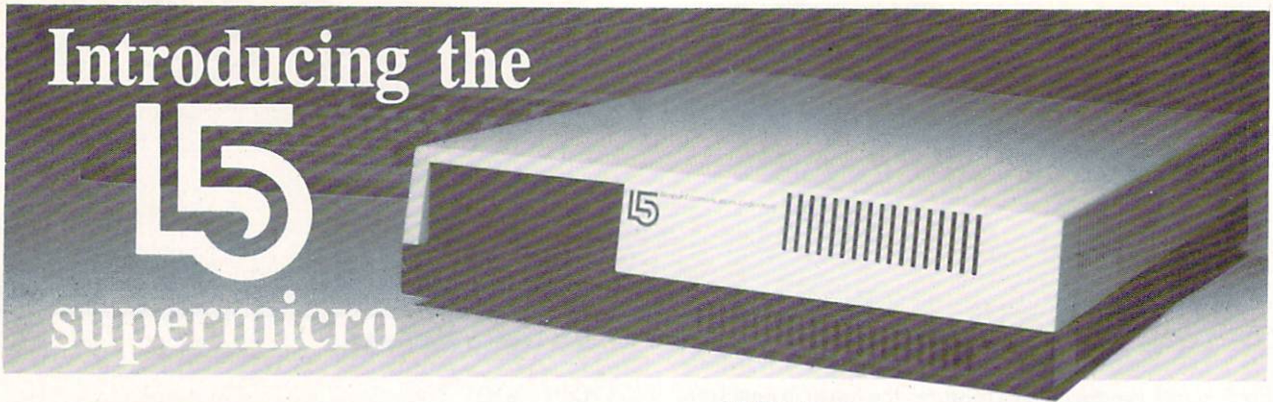
**MTBASIC**, a multitasking BASIC compiler, promises fast, highly optimized object code and many advanced features.

Up to 10 windows can be created on the screen simultaneously, assigned to separate tasks, or overlayed to implement pop-up and pop-down menus. Multitasking allows up to 10 programs to run concurrently. The ability to start tasks upon receipt of hardware interrupts facilitates real-time systems control. High-level hooks are provided for the addition of user I/O devices.

The MTBASIC compiler runs interactively and is said to compile more than 100 BASIC statements per second. MTBASIC is available from **Softaid Inc.**, runs under PC-DOS or CP/M 80, and retails for $49.95.

Softaid Inc., P.O. Box 2412, Columbia, Md. 21045-1412, (301) 792-8096.

**CIRCLE 104 ON READER SERVICE CARD**



**A breakthrough in Price, Performance and Packaging**

The new L5 proves that good things come in small packages! Measuring a compact 3.75"H × 17.5"W × 21"D, the L5 is small enough to fit on a desk-top or a laboratory workbench. Yet it's large enough to handle up to 32 users and, in some applications, outperform a VAX system. The L5 is small in price, too. A mid-range system can cost less than $1,000 per user!

Find out how the new L5 offers the unique solution to price, performance & placement challenges. Call General Communications today!

**General Communications Corporation**

*"Where lucid communications, technical excellence and common sense meet."*

*1 Main Street, Suite 502, Eatontown, NJ 07724 (201) 542-6560*

L5 Features

- KDJ11 Processor, floating point, 8K cache
- .5Mb to 32Mb memory
- 4 to 32 users
- 20Mb to 2.5Gb external storage
- UNIX System V Fast Kernel or Real Time Kernel
- Runs RT11, RSX, and TSX
- Includes several utilities packages, necessary cabling, complete documentation and tutorials

***Aggressive Dealer/OEM discounts available***

*UNIX is a trademark of AT&T-Bell Laboratories
VAX, RT11, & RSX are trademarks of Digital Equipment Corporation
TSX is a trademark of S&H Computer

**CIRCLE 60 ON READER SERVICE CARD**

MODULA-2 PROGRAMMING TOOLS

A collection of utility modules ready to link into your programs and greatly speed programming efforts and the operation of programs.

Each tool is supplied as a definition module with in-line documentation, an implementation module with full source code and a ready-to-link object module. A fully-linked ready-to-run test program with source code is included.

Each module is implemented using Logitech's Modula-2/86™, Version 1.1 and MS-DOS/PC-DOS™ Version 2.0 or later unless otherwise specified. All modules are upward compatible with Microsoft's Xenix™ operating system as specified in the Microsoft MS-DOS Programmer's Reference Manual.

**MemUtils**: high-speed memory utilities coded using 8086 string instructions.
**Keyboard**: a complete IBM-PC keyboard handler.
**ScreenOps**: high-speed routines for controlling IBM-PC text screen. Based on ROM BIOS calls.
**FileOps**: direct access to MS-DOS file handling functions via DOS function calls.
**DirOps**: direct access to MS-DOS's hierarchial directories via DOS function calls.
**DiskUtils**: miscellaneous disk and drive utilities via MS-DOS function calls.
**SingVD**: calculates singular values of real-values matricies.
**MicroMouse**: direct access to all 16 Microsoft Mouse funtcions via mouse system software function calls.

| | | | |
|---|---|---|---|
| Memutils | $29 | All three for $59 | Entire package of 8 modules - all with source code and test programs for $189 |
| Keyboard | $39 | | |
| ScreenOps | $39 | | |
| FileOps | $39 | All three for $79 | |
| DirOps | $39 | | |
| DiskUtils | $29 | | |
| MicroMouse | $49 | | |
| SingVD | $89 | | |

Developed by: Thomas H. Woteki, Ph.D.

Add $3/order shipping and handling    VA residents add 4% sales tax

Call 703/ 522-8898 or send your order to: **Information Systems Incorporated**
1901 No. Fort Myer Drive, Arlington, VA 22209

-Quality Software At Low Prices-    -Save Time With Expert Tools-

**CIRCLE 20 ON READER SERVICE CARD**

# SOFTWARE REVIEW

## Roundup of 29 BASIC interpreters and compilers

**By Namir Clement Shammas, Steve Leibson, Jay Halcomb, and Stephen Martin**

A famous saying goes like this: "Don't kick a man when he is down, he might get up." After plenty of criticism, BASIC is "getting up." This review invites you not only to compare BASIC packages but to see how BASIC has evolved.

The BASIC programming language was born over two decades ago at Dartmouth College. It took over a decade for it to find its niche in the world of microprocessors and microcomputers. Today almost every microcomputer comes with an implementation of BASIC. For many who want to learn programming, its ease of use has made it the first language chosen. Yet BASIC has had its share of competition and criticism. Some say that it invites the programmer to write spaghetti code riddled with *GOTO* patches. Even the creators of BASIC, John Kemeny and Thomas Kurtz, recently commented that BASIC implementations for micros have deviated from the original BASIC. The creators of BASIC followed their words with action: their True BASIC conforms to the proposed ANSI BASIC.

BASIC on micros has come a long way. We still remember the PET Commodore, Radio Shack, and Apple II micros with their primitive BASICs. Since then, Microsoft BASIC has emerged as the industry's de facto standard. It is available for all popular micros. Its superset implementation, IBM PC's BASICA, is also popular.

Other BASIC implementations have also emerged, some offering more sophisticated language aspects. This is seen by some professionals as a strong influence of the Pascal language on BASIC.

In this review we will be looking at BASIC implementations for microcomputers running MS-DOS/PC-DOS, CP/M 86, CP/M, and the Apple Macintosh Finder. An IBM PC with the 8087 chip was used for the first two. A Kaypro II was used to test the CP/M BASIC.

The five tables in this review include the product's general information, Table 1; data types and variables, Table 2; the program flow control, decision making and error handling, Table 3; functions, subroutines, and overlays, Table 4; and file I/O and graphics, Table 5.

Four benchmark test programs were used to compare the speed of execution (see sidebar). They are:

- The Sieve of Eratosthenes
- A test that reverses the sorted order of an array with 1,000 integers
- The Pournelle matrix test
- The sine/cosine test

### MS-DOS INTERPRETERS
### American Planning Corp. MEGABASIC

This interpreter has evolved over the past several years. American Planning Corp. originally wrote APC BASIC to use under CP/M on NorthStar computers. With the introduction of 8088-based machines, APC BASIC has become APC MEGABASIC. The MEGA refers to the ability of the interpreter to use the full 1MB address space of the processor and to its powerful capabilities.

MEGABASIC is a full-featured interpreter designed to run on a full range of machines. It is not limited to the IBM PC. It offers most of the features of advanced BASIC interpreters, including long variable names (250 characters) and extensive program control structures. MEGABASIC is somewhat unusual in that it offers BCD arithmetic instead of binary. The accepted myth is that BCD arithmetic is slower, although that does not show in the benchmarks.

The outstanding features of this interpreter make the product very notable. Integers have been added to this language starting with version 5.0 to allow speedy coding, especially in loop counters. These are 32-bit integers, a departure from most other BASICs, which offer 16-bit integers.

Memory can be divided into as many as 64 workspaces, each with its own program or function library. Programs can access functions, procedures, and programs in other workspaces through a rigorous access protocol. This allows your program to use as much memory as required. Most BASIC language implementations, even in the 8088 world, limit code to 64K. Each array or string can use up to 64K, allowing large data structures as well as large program capability.

The workspace concept should not be overlooked. It allows the programmer to create function libraries similar to C libraries to be used in later programs. This is the basis for creating software productivity tools.

One reason BASIC is so popular is that it is simple to try out an idea quickly. In MEGABASIC it is even easier. Whenever you see the interpreter prompt, typing "BASIC" will invoke a nested copy of the interpreter, isolating you from your working program and providing you with a clean slate with which to experiment.

### Control-C BASIC Interpreter 286

This implementation from Control-C Software Inc. differs greatly from Microsoft BASIC. New commands have been added— some renamed and implemented differently—while others, like the trigonometric, transcendental and square root functions, are completely eliminated.

The BASIC Interpreter uses a librarian to add a new file, remove an old file, compact dead spaces, display the directory, and write library files to disk. BI has a line editor to alter program lines. It has a range of 1 to 9,999, smaller than what MS-BASIC allows.

BI has two data types: numeric and strings. The *PRECISION* statement sets the number of decimal places, ranging from zero to seven, for the output, operation, and comparison of numeric expressions and variables. Variable names are limited to one character followed by one digit. This hampers program readability and maintenance.

Dimensioning strings signifies assigning the maximum number of characters per single string. A string may be initialized while being dimensioned. This comes in the form of filling it with a string literal, another string variable or string expression.

The only loop construct is the *FOR-NEXT* loop, with an *EXITTO <line number>* option for exiting the loop to a specific line number. Decision making is carried out with the *IF* statement. The *THEN* keyword is dropped. Instead

*GOTO*, *GOSUB* or other statements can be written after the logical test. No *ELSE* clause is available. *ON GOTO* and *ON GOSUB* statements are available.

BI offers one-line, user-defined functions. Subroutines are implemented similarly to those in MS-BASIC (that is, called by line number). BI offers a number of built-in functions to perform character and numeric conversions and manipulations. The *STR* function is noted for its ability to convert numbers into formatted strings. This can be used with *PRINT* statements to output formatted numeric data, as *PRINT USING* in MS-BASIC and most other BASIC implementations.

BI can *CALL* public programs and pass arguments. The concept of public programs comes from the creation of commonly shared program/routines in a multiuser environment. The BI librarian, mentioned earlier, is responsible for managing the program library.

BI implements advanced data file organization. It is the main strength of the language. Data can be stored in serial (that is, sequential), direct, indexed, and sort files. Serial files contain a simple sequence of stored data records of varying lengths. All other data file types use fixed-length records and support their random access. Indexed files have their data stored sequentially. Information can be accessed sequentially or by using an index. Direct files are similar to the indexed ones, except an additional B-tree structured directory exists. This facilitates more efficient searching.

Sort files are sort-key files that maintain an ascending order. BI offers versatile I/O statements. *INPUT*, *PRINT*, *READ*, and *WRITE* statements offer many powerful options such as error handling, setting screen cursor position, and issue prompts.

The benchmark tests reflect the slowness and the limitations of BI. The Sieve test was not carried out because of the inability to dimension an array with 8,191 elements. BI's lack of trig functions prohibited the trig test. The sorting and matrix test results put it among the slowest interpreters.

### Microsoft IBM PC BASIC (BASICA) and MS-BASIC

The BASICA implementation was developed by Microsoft Corp. for the IBM PC. It is a superset of MS-BASIC. BASICA in turn has also become popular due to its downward compatibility with MS-BASIC and the large number of IBM PC microcomputers used. The major difference between BASICA and MS-BASIC is the latter's lack of graphics capability. Otherwise, the two dialects are identical.

BASICA and MS-BASIC are true interpreters. Each program line must start with a number. BASICA has a screen editor available to change lines of code, while MS-BASIC offers a line editor.

Four data types exist: integer, single and double precision reals, and strings. The four symbols used are %, !, #, and $, which can be appended to a variable name and reflect the data type contained. BASICA and MS-BASIC allow arrays to be erased and redeclared. This provides a method of practically redimensioning an array. The catch is that the data in the old array must be saved somewhere.

Loops consist of the *FOR-NEXT* and *WHILE-END*. Decision making employs *IF-THEN* with an optional *ELSE* clause. The entire *IF* construct must be located on one program line.

BASICA and MS-BASIC allow for single-line, user-definable functions. They are allowed to have multiple arguments and return a value in any of the four data types. Subroutines are called by a *GOSUB* statement followed by a line number.

With regard to chaining programs, BASICA and MS-BASIC offer very flexible and powerful features. First, it is very easy to pass all the variables to the chained program. You can also specify which line number in the chained program

**If you can't share files on PC Network, you're using the wrong file manager.**

**Be connected. Btrieve.™**

Networks can solve problems. But running a single-user file manager can create new ones: Lost updates. Garbled data. Trashed files.

Btrieve™/N offers *safe* multi-user file management that protects your data when sharing files. And eliminates the need to rewrite your application for LANs. Btrieve/N set the file management standard for the industry's most popular networks: Netware, Davong MultiLink, Omninet, PC Net, EtherSeries, Nestar, and NetOne. And now IBM's PC Network.

**Fast.** Btrieve/N is fast, too. It's written in assembly language especially for the IBM PC. And based on b-tree file indexing, for access speed that won't degrade as your database grows.

**Automatic file recovery.** Btrieve/N provides automatic file recovery after a system crash. Your Btrieve data always comes back intact.

**Fully-relational data management.** SoftCraft's entire family of products gives you a complete, fully-relational database management system. Rtrieve™/N adds report writing capabilities. Xtrieve™/N speeds users through database queries with interactive menus.

**For professional programmers.** Btrieve/N is the fast, reliable answer for all your application development in BASIC, Pascal, COBOL, C, FORTRAN, and APL. With Btrieve/N, you can develop better network applications. And solve problems, not create new ones.

SoftCraft Inc.
P. O. Box #917 Austin, Texas 78766
(512) 346-8380 Telex 358 200

*Suggested retail prices: Btrieve, $245; Btrieve/N, $595; Xtrieve, $195; Xtrieve/N, $395; Rtrieve, $85; Rtrieve/N, $175. Requires PC-DOS or MS-DOS 1.X, 2.X, or 3.X.*

CIRCLE 38 ON READER SERVICE CARD

## Benchmarks

The review team used four benchmark tests—Sieve of Eratosthenes, array sorting, matrix, and sine/cosine—to examine the speed of the BASIC packages while performing a variety of tasks. The aim was to attempt running the same listings without having to resort to any helpful features particular to any language. It was felt that this would make the review more fair and accurate. Many versions required minor changes in items like variable names, but the general structure of the programs remained consistent.

We used the popular Sieve of Eratosthenes test so readers could compare the timings we obtained with those published elsewhere for other languages.

The array sorting test creates an integer array with 1,000 members whose values are in perfect ascending order. The test examines the speed with which the order of values is reversed. This test reflects the speed of handling integers and sorting in memory.

The matrix test, based on one developed by Jerry Pournelle (*BYTE*, October 1982), reflects the speed of handling floating points in general and matrices in particular. Three matrices are initialized and two of them are multiplied, storing the result in the third. The sum of all the elements in the third is also calculated. The results are very significant for scientific, engineering, and financial calculations.

A trigonometric test examines the speed of calculating sines and cosines. It is important for calculations involving trig functions, such as two- and three-dimensional graphics packages. The results are indeed interesting. The test involves calculating each sine and cosine 720 times.

SuperSoft Programmer Utilities
When Performance Counts

Star-Edit and Disk-Edit

**Star-Edit is the professional programmer's text editor** with an outstanding list of commands tailored to program development. It can greatly simplify all your editing tasks – moving and reproducing text or code, viewing two files simultaneously through separate windows, moving text or code between different files, searching forward or backward, and moving to the the beginning or end of any word, sentence, paragraph, parentheses, or curly brackets. Virtual memory makes Star-Edit ideal for extremely large files; and because it never uses over 128K, it is well suited for multiple process and windowing environments. (PC DOS, MS DOS, CP/M-86, CP/M-80, UNIX, or XENIX): $225.00

To order call: **800-762-6629**
In Illinois call: **217-359-2112**
or write to **SuperSoft**.

**Disk-Edit is the uniquely powerful disk utility for programmers** which gives you access to every bit of information on your disk. It lets you read disk data in both HEX and ASCII, "text edit" any information on your disk, restructure disk information, and save lost or scrambled data. Imagine scrolling through your disk data, jumping between HEX and ASCII windows, and editing information anywhere on your disk. For all floppy and hard disk systems. (PC DOS, MS DOS, CP/M-86, CP/M-80, UNIX, or XENIX): $100

SuperSoft

1713 S. Neil St., P.O. Box 1628
Champaign, IL 61820
telex:270365

CIRCLE 79 ON READER SERVICE CARD

A FULL C COMPILER FOR $49.95

The Ecosoft Eco-C88 compiler for the 8088 and MSDOS is going to set a new standard for price and performance. Consider the evidence:

| Compiler | Eco-C88 | Latice (1) | C86 (1) |
|---|---|---|---|
| Seive | 13 | 11 | 13 |
| Fib | 44 | 58 | 46 |
| Deref | 13 | 13 | -- |
| Matrix | 21 | 29 | 27 |
| Price | $49.95 | $500.00 | $395.00 |

(1) **Computer Language,** Feb., 1985, pp.73-102. Reprinted by permission.

The Eco-C88 compiler is a full K&R C compiler that supports all data types and operators (except bit fields). Now look at the other features we offer:

★ 8087 co-processor support using a single library. If you install an 8087 later, the software will use it without having to recompile.
★ A robust standard library with over 150 functions, including transcendentals, color, and others.
★ OBJ output for linking with the MSDOS linker (LINK).
★ Error messages in English – no cryptic numbers to look up. A real plus especially if you're just getting started with C.
★ Easy-to-read and complete user's manual.
★ Works with all IBM and compatibles running MSDOS 2.0 (or later).
★ Plus many other features.

For $10.00 more, we will include the source code for the C library functions (excluding transcendentals). For an additional $15.00, we will include our ISAM file handler in OBJ format (as published in the **C Programmer's Library,** Que Publishing). The discount prices for the library source and ISAM only apply at the time the compiler is purchased. Please add $4.00 to cover postage and handling. To order, call or write:

ECOSOFT INC.

**Ecosoft Inc.**
6413 N. College Avenue
Indianapolis, IN 46220
(317) 255-6476

MasterCard VISA AMERICAN EXPRESS

Eco-C (Ecosoft), MSDOS (Microsoft), UNIX (Bell Labs), CP/M (Digital Research), Z80 (Zilog), 8086, 8087, 8088 (Intel).

CIRCLE 17 ON READER SERVICE CARD

will be executed first. Using the *CHAIN MERGE* option, you can dynamically alter a running program—a powerful and versatile technique.

BASICA has both medium- and high-resolution graphics. The user is presented with many versatile commands. A program can define the screen portion to be used for graphics and the real-world scales for the *X* and *Y* axis. BASICA can plot points, draw lines and shapes, fill in shapes, mix text and graphics and has many more commands to accommodate animation and sophisticated graphics.

File I/O allows sequential and random access of records. Carrying out the latter is not straightforward. First, the *FIELD* declaration defines buffer variables, all of which are strings, for the set of fields involved. These variables are then assigned the information to be stored.

BASICA and MS-BASIC provide functions to convert integers and reals into coded strings and vice versa. This frees the programmer from having to worry about the number of digits to be stored as well as the imminent loss of precision.

Error handling employs the traditional *ON ERROR GOTO <line number>* statement, coupled with a *RESUME* statement to direct the optional reassumption of execution.

The benchmark test shows the BASICA running slightly faster than MS-BASIC. Overall, they are rated as slow since they do not semicompile their programs.

### Morgan Computing Professional BASIC

This excellent product comes to us from Morgan Computing Co. Inc. In a nutshell, it is an IBM PC BASIC subset/superset with extensive and very powerful tracing and debugging capabilities (see review of version 1.0 in the premier issue of *COMPUTER LANGUAGE*). Professional BASIC is semicompiled and thus runs faster. It performs dynamic syntax checking on each program line as it is entered. New commands are implemented to perform versatile editing and cross referencing of variables. This includes commands to find variables, lines, and labels, search for a specific text, and display a sorted list of variable names and labels.

Tracing program execution is essentially what Professional BASIC is all about. Different types of tracing windows exist. While running a program in single-step mode and examining different windows, you can view the contents of simple variables, arrays, and file buffers. In addition, you can trace the execution of program lines, *FOR-NEXT* loops, and subroutines. One window even displays the number of times each program line has been executed. These windows help to pinpoint any problem you may have with a program, without wasting a lot of time.

Professional BASIC uses four bytes to store integers, compared with the two bytes used by BASICA. This allows all the integers in the former to be double precision and the storage of numbers between plus and minus 2.147 billion. Double precision reals have an extended range too, with a maximum exponent of plus/minus 308. All arrays must be defined; no default size is assumed. The lower value of an array subscript can be other than one or zero. Thus *DIM Year(1950 to 1979)* defines a 30-element array with a lower subscript of 1950. The upper limit of the subscript can be as large as 2 billion! Only integer constants are allowed in dimensioning arrays.

Professional BASIC has a unique feature for keying in long variable names. To avoid misspelling them, the user keys in a minimum number of letters to uniquely identify the variable and then enters @, causing the rest of the name to be displayed.

Loops consist of *FOR-NEXT* and *WHILE-WEND*. Professional BASIC requires that *FOR* and *NEXT* statements be physically and logically paired. The same applies for *WHILE* and *WEND* statements. *EXITFOR* and *EXITWHILE* statements have been added to exit *FOR* and

PROGRAMMING TOOLS
for Pascal and C

**Pascal Support**

Microsoft (IBM)Pascal is an effective tool for serious application development. Powerful abstract data capabilities, block structure, modular compilation, and direct memory addressing produce code that runs fast, and is easy to maintain. The Blaise Computing Programmer Productivity Series makes this environment even more attractive and speeds completion of your systems. You can obtain these tools from us for a fraction of the cost of developing them yourself. Source code is either included or available with each package, so you can make modifications easily. No royalties are required for incorporating our routines into your systems.

Blaise Computing's Pascal products have been reviewed favorably many times during their two years on the market, and in a recent survey, our users report them to have been extraordinarily valuable.

- **Pascal TOOLS**™ is a library of procedures including powerful string manipulation, screen and keyboard handling, and graphics primitives. There are also a command line parser, a general BIOS gate for easy use of all hardware services, and many other features. **$125**

*Develop your Applications Quickly*

- **Pascal TOOLS 2**™ gives access to the advanced operating system services of DOS 2.0 and later versions. Your Pascal program can retrieve DOS system information, perform full memory management, and execute other programs. There is a general gate to access all DOS services, and specific procedures are provided for handling files, directories, and disk I/O. **$100**
- **VIEW MANAGER**™ is our display screen management system that makes screen development and documentation much faster. A complete library of Pascal procedures implements block mode data entry and transmission. Information can be captured or recalled, and context-relevant help files displayed with simple procedure calls. **$275**
- **ASYNCH MANAGER**™ is a library of interrupt-driven routines providing a general interface to both COM ports for your asynchronous communications applications. Introductory price of **$175** includes all source.

*Blaise Computing's Programmer Productivity Series is also available in versions for the C language.*

**BLAISE COMPUTING INC.**
2034 Blake Street Berkeley, CA 94704
(415) 540-5441

CIRCLE 8 ON READER SERVICE CARD

*WHILE* loops, respectively. Decision making is performed using the one-line *IF-THEN* construct with an optional *ELSE* clause.

Data files are implemented similarly to those in BASICA. However, Professional BASIC uses unsigned single and double precision integers. Thus a file may contain slightly over 4 billion records or bytes. A record can hold 65,535 bytes.

The benchmark tests indicate that the use of the 8087 version is desirable when performing floating point math. Without the use of the 8087 chip, Professional BASIC becomes about four times slower than MS-BASIC!

### Ryan-McFarland RM-BASIC

Ryan-McFarland is known for its COBOL packages for microcomputers, and RM-BASIC is capable of reading data files written by RM-COBOL and RM-FORTRAN. RM-BASIC has powerful line editing commands that operate on a range of program lines instead of being limited to single lines. They allow search and find/replace variables and text. RM-BASIC also permits the user to execute an MS-DOS command using the *SYSTEM <DOS command>* statement.

RM-BASIC has integer, 15-digit real, and string data types. Dimensioning strings may include declaring the maximum string size, whose default value is 18. RM-BASIC can handle matrix math operations, allowing matrix processing without using explicit loops. This includes the ability to redimension the matrix and establish ascending/descending index matrices that reflect the order in other matrices. Unfortunately, no statements for matrix inversion, determinant value, or solving a system of linear equations are available.

RM-BASIC offer the *FOR-NEXT* as the only loop construct. Decision making involves the single line *IF-THEN-ELSE*. RM-BASIC allows for labels to be used in directing *GOTO* jumps and *GOSUB*s.

Multiline, nonrecursive, user-definable functions are allowed. String type functions may even declare the maximum size of the string returned. Variables that appear inside the function body are considered global if they are not declared in the function's argument list. Subroutines are *GOSUB*ed using labels or line numbers. External assembly language subroutines can be *CALL*ed.

When it comes to file I/O, RM-BASIC really flexes its muscles. There are three types of file structures: display (ASCII format only), native (ASCII and binary format), and internal (binary format only). Files are organized as sequential, relative, or keyed.

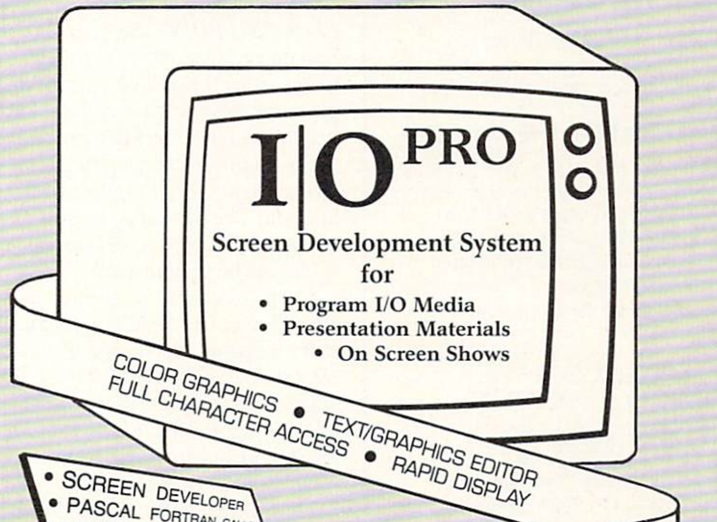
Sequential files have their data written to and read from serially. Relative files access their records using record numbers. Keyed files use a two-file organization system: one for data, the other for keys. This is used in organizing native typed data files only. RM-BASIC has a number of powerful and versatile statements for record manipulation and management.

RM-BASIC has high-resolution graphics statements, including color selection, plotting points, drawing lines, polygons, boxes, circles and arcs, and mixing text with graphics. Graphic shapes can also be *FILL*ed.

Error handling constructs are very powerful. The error handling statement uses error condition keywords to specify the type of error to be trapped. The action taken may be either a *GOTO* to resume execution or displaying an error message and halting the program. In addition, the *EXIT* statement can be used in conjunction with a list of instruction pairs, each containing an error code number with the accompanying line number. Each line number directs the program to the appropriate error handling section.

Errors can also be simulated using the *CAUSE* statement. RM-BASIC provides

I/O A BORE? NOT ANY MORE!


I|O PRO
Screen Development System
for
• Program I/O Media
• Presentation Materials
• On Screen Shows
COLOR GRAPHICS • TEXT/GRAPHICS EDITOR
FULL CHARACTER ACCESS • RAPID DISPLAY
• SCREEN DEVELOPER
• PASCAL FORTRAN CALLS
• DISPLAY PROGRAMS
• GRAPHICS LIBRARY
COMPLETE I/O FIELD DEFINITION
FORMAT • RANGE • ALLOWED RESPONSES
DEFAULTS
IBM OR PLANTRONICS COLOR/GRAPHICS BOARD
MS OR PC-DOS 192K RAM
ARTIFICIAL INTELLIGENCE • CUSTOM DATA ENTRY
REPORT FORMATS • ON SCREEN/HARDCOPY PRESENTATIONS
ENHANCE PROGRAM INTERACTION
ELIMINATE SCREEN ENTRY ERRORS
SLASH DEVELOPMENT COSTS
REVISE SCREENS IN SECONDS

**M|E|F Environmental, Inc.**
P.O. Box 26537
Austin, TX 78755 (512) 251-5543

$450 : OEM Pricing and Licenses Available
Demonstration Diskette $10
applicable to purchase

CIRCLE 15 ON READER SERVICE CARD

C

SOFTWARE
DEVELOPERS!

V - FILE
THE VIRTUAL MEMORY
FILE MANAGER

Let V-FILE save precious development time & cost as you create efficient applications with the power of VIRTUAL MEMORY.

DON'T RE-INVENT THE WHEEL

Why spend weeks or months coding and debugging file and memory management systems when you can order V-FILE today. V-FILE is a library that you can link with your code to provide sophisticated virtual file and memory management — allowing you to concentrate on developing your application.

VIRTUAL DATA
OBJECTS SUPPORTED!

Data is referenced by using VIRTUAL MEMORY DATA HANDLES. Your code doesn't need to know whether the data is actually on disk or in RAM. Swapping between disk and RAM and updating files on disk is handled automatically and transparently! Complex VIRTUAL DATA STRUCTURES can be created by linking with data handles instead of pointers.

CHECK THESE FEATURES!

- Multiple, independent swap buffers
- Multiple files per swap buffer
- Highly efficient swap algorithm
- Automatic file updating
- Data prefetching supported
- Data may be locked in memory
- Memory buffers may be flushed
- Makes full use of extended memory on IBM PC/AT
- SOURCE CODE AVAILABLE
- NO ROYALTIES REQUIRED

Supports Dos 2.00+ with
Lattice & Microsoft C compilers
Supports Microsoft windows

$299

V-FILE
TM

Contact:
MindBank, Inc.
4620 Henry Street
Pittsburgh, PA 15213
412/683-9800

VISA/MASTER CARD ACCEPTED

CIRCLE 63 ON READER SERVICE CARD

functions that return various information regarding the type and location of the errors.

### Southwest Data Pluto BASIC

This interpreter from Southwest Data Systems is very similar to the BI dialect. The implementation we looked at is a microcomputer version of a BASIC dialect used in minicomputers. It is compatible with Science Management Corp. (SMC) BASIC and BASIC/Four BOSS levels 3 and 4. This MS-DOS version supports file directory paths and allows DOS commands to be issued from the interpreter.

Pluto BASIC's data types are numeric and character strings. No explicit distinction exists between variables that contain integers or floating point reals. The *PRECISION* statement is used to set the number of decimals between 0 and 14.

To maintain full floating point the *FLOATING POINT* statement is used. Simple variables have their names limited to one letter followed by an optional single letter. Numeric array names are limited to a single letter. Dimensioning a string implies declaring the size of one string. Strings are initialized during declaration. The default is the ASCII space character. String expressions and constants can be used instead.

Loop constructs are limited to the *FOR-NEXT* loop. The *EXITTO* statement can exit a loop and subroutine. Decision making uses the multiline *IF-THEN-ELSE*, which employs the colon symbol after a line number to signal continuation. Multiple statements are separated by semicolons.

Pluto BASIC offers a good number of functions and procedures that are new to MS-BASIC programmers. This includes functions for conversions among strings and decimal, hexadecimal, and binary numbers. Scientific functions are also implemented. This includes the square root, base ten and natural logarithms, exponent, and trigonometric functions.

Functions for error handling and management, day and time, string manipulation, random number generation, bit manipulation, and file inquiries exist. Pluto BASIC offers the *EXECUTE* statement, which takes a string and passes it to the interpreter to be executed as a program line. User-definable functions are limited to one-line functions. Their names are limited to one letter, followed by an optional question mark. The dollar sign can be used for functions that return strings. Internal subroutines are called with a *GOSUB* followed by a line number. Pluto BASIC can *CALL* public programs and pass arguments. Such routines must be *ADD*ed into the System File Table which keeps tracks of such programs. Conversely, they can also be *DROP*ed (removed from the list). The *EXIT* statement terminates the execution of a public program and returns control to the calling program.

File I/O in Pluto BASIC is very similar to that of Control-C Software's BI dialect. The types of data files are the program, serial, indexed, direct, and sort. Pluto BASIC offers very powerful and flexible file, printer, and console I/O commands with excellent imbedded error trapping.

The reviewed version Pluto BASIC seems to be the first to implement extended video and graphics control. The *SYS* statement is used to send arguments to the IBM PC ROM BIOS and perform a variety of functions including a video type and mode check, selection of video pages, controlling character foreground and background attributes, setting the color, selection plotting coordinates, and plotting points. Using the *SYS* statement as a multipurpose command leaves much to be desired. Distinct commands are needed to enhance the clarity of graphics routines.

Error handling is similar to that of the BI package and is handled by using error clauses with a variety of statements. This gives maximum control over unexpected events.

Pluto BASIC was only able to run the trigonometric benchmark tests. The interpreter flagged an out-of-allocated-memory for the other test programs that had array sizes larger than the allocated data space per program.

Pluto BASIC passed the trig test—the only one. While its trig functions are slower than most other interpreters, they are faster than Better BASIC.

### Summit Software Better BASIC

This product from Summit Software Technology Inc. is a new implementation which brings forth many powerful, practical, and versatile features and enhancements to programming in BASIC. It could be renamed Much Better BASIC without exaggeration. Better BASIC uses a user-adjustable configuration file, which can alter operating parameters, such as numeric precision, stack height, and size of extended memory. The configuration file also contains the names of modules that extend the capabilities of Better BASIC. The distribution diskette comes with a number of mandatory and optional modules. User-created modules can also be included. Better BASIC provides a screen editor to alter programs.

Better BASIC allows advanced and sophisticated data types and variables. Among the reviewed BASIC implementations running under MS-DOS, we rated it number one for data types. Better BASIC has the following basic data types: integer, byte (0 . . . 255), real, string, and pointer (to any other type). Structured

# Add EDITING to your Software CSE Run-Time™

Your program can include all or a portion of the *C Screen Editor* (CSE).

CSE includes all of the basics of full screen editing plus source in C for only $75. For only $100 more get CSE Run-Time to cover the first 50 copies that you distribute.

Use capabilities like Full cursor control, block move, insert, search/replace or others. Portability is high for OSes, terminals, and source code.

Call for the ''CSE Technical Description'' and for licensing terms and restrictions.

**Full Refund if** not satisfied in first 30 days.
**Call 800-821-2492**

Solution Systems™
335-L Washington Street
Norwell, MA 02061
617-659-1571

CIRCLE 37 ON READER SERVICE CARD

---

**PROFESSIONAL PROGRAMMER'S BULLETIN:**

# Be Productive, Be BRIEF™

The Programmer's Editor

TRY BRIEF "RISK-FREE" FOR 30 DAYS WITH OUR MONEY-BACK GUARANTEE!

BRIEF's power and flexibility provide dramatic increases in programming productivity. BRIEF's ergonomically designed human interface becomes a natural extension of your mind, allowing you to eliminate tedium and concentrate on creativity.

- WINDOWS
- Full UNDO (N Times)
- Compile within BRIEF
- Keystroke Macros
- Exit to DOS inside BRIEF
- Programmable Macro Language
- Multiple files, unlimited size
- "Regular Expression" search
- Reconfigure keyboard
- Language sensitive user controllable features (such as Auto-Indent for C)

AVAILABLE FOR PC-DOS, IBM-AT, AND COMPATIBLE SYSTEMS

ONLY $195.

DEMO AVAILABLE FOR ONLY $10 (applicable to future purchase)

CALL TOLL FREE
800-821-2492
for "Technical Description" or to order.

Solution Systems™
335-L Washington St., Norwell, MA 02061
617-659-1571

BRIEF is a trademark of UnderWare.
Solution Systems is a trademark of Solution Systems.

CIRCLE 88 ON READER SERVICE CARD

---

# C Helper™

FIRST-AID FOR C PROGRAMS

Save time and frustration when analyzing and manipulating C programs. Use C HELPER's UNIX-like utilities which include:

**DIFF** and **CMP** – for "intelligent" file comparisons.
**XREF** – cross references variables by function and line.
**C Flow Chart** – shows what functions call each other.
**C Beautifier** – make source more regular and readable.
**GREP** – search for sophisticated patterns in text.

There are several other utilities that help with converting from one C compiler to another and with printing programs.

C Helper is written in portable C and includes both full source code and executable files for $135 for MS-DOS, IBM AT CPM-80 or CPM-86. Use VISA, Master Card or COD.

Call: **800-821-2492**

Solution Systems™
335 Washington Street
Norwell, MA 02061
617-659-1571

CIRCLE 89 ON READER SERVICE CARD

---

# PROLOG-86™

## Become Familiar in One Evening

Thorough tutorials are designed to help learn the PROLOG language quickly. The interactive PROLOG-86 Interpreter gives immediate feedback. In a few hours you will begin to feel comfortable with it. In a few days you are likely to know enough to modify some of the more sophisticated sample programs.

## Sample Programs are Included like:

- an EXPERT SYSTEM
- a NATURAL LANGUAGE INTERFACE (it generates a dBASE II "DISPLAY" command)
- a GAME (it takes less than 1 page of PROLOG-86)

## PROTOTYPE Ideas and Applications QUICKLY

1 or 2 pages of PROLOG is often equivalent to 10 or 15 pages in "C" or PASCAL. It is a different way of thinking.

Describe the FACTS and RULES without concern for what the computer will have to do. Maybe you will rewrite in another programming language when you are done.

**Programming Experience is not required** but a logical mind is. PROLOG-86 supports the de facto STANDARD established in "Programming in Prolog."

**AVAILABILITY:** PROLOG-86 runs on MSDOS, PCDOS, IBM AT or CPM-86 machines. We provide most formats. The price of PROLOG-86 is **only $125.**

Full Refund if not satisfied during first 30 days.
800-821-2492

Solution Systems™
335-L Washington Street
Norwell, MA 02061
617-659-1571

CIRCLE 90 ON READER SERVICE CARD

**RUN/C:™**
***Finally,***
**a C Interpreter**

Available NOW for only **$149.95!**

Finally, a painless introduction to the C language. With **RUN/C: The C Interpreter** you can create and run C language programs in an environment as easy to use as BASIC.

**RUN/C** is C for the rest of us. It is a robust implementation of standard K&R. **RUN/C** is for both the beginner and professional.

**RUN/C** includes full floating point, 8087 support, structures, unions, casts and more than 100 built-in C functions.

With **RUN/C** you get all this with a command structure modeled after BASIC's using familiar terms such as EDIT, RUN, LIST, LOAD, SAVE, TRON, SYSTEM, etc.

Since **RUN/C** is a true interpreter it means that C programs can be written, tested and run within a single protected environment. It is a teaching tool and a source code debugger.

Here's more good news . . .

- Great documentation: a 400-page, easy-to-read manual filled with executable programs
- Array-index and pointer bounds checking
- Variable-trace and dump diagnostics PLUS an integral program profiler
- Full buffered and unbuffered file I/O
- Printer and asynch support
- Forking to your favorite full screen editor with automatic return to **RUN/C** with your edited program
- System Requirements: IBM® PC or compatible with PC-DOS 2.0 or MS™-DOS 2.0 or greater with ANSI.SYS.

Get things right the first time with **RUN/C: The C Interpreter.™**

For immediate delivery or more information, call:

**1-800-847-7078**
(in N.Y. **1-212-860-0300**)

or write: Lifeboat Associates™
1651 Third Avenue
New York, NY 10128

RUN/C is a trademark of Age of Reason Co.

**CIRCLE 87 ON READER SERVICE CARD**

records can be defined using these or other previously defined records.

Loop constructs are abundant in Better BASIC. In addition to the *FOR-NEXT* loop, many other types of loops serve different purposes. First, the *DO <x> TIMES* loop is similar to the *FOR-NEXT* loop. *WHILE <cond> -DO* and *DO IF <cond>* loops test a condition before executing a block of statements. The former will execute the block as long as the condition tested is true.

*DO IF* will execute the block once at most. *DO UNTIL <cond>* and *DO-REPEAT IF <cond>* will execute a loop at least once and repeat if the tested condition is true. *DO-END DO* and *DO-REPEAT* open loops are also available. Better BASIC allows the user to *EXIT* from all loop types. In addition to exiting the current loop, *EXIT <n> LEVELS* allows exiting from a number of nested loops. Decision making is limited to the single-line *IF-THEN-ELSE* construct.

Better BASIC offers many built-in numeric and string functions. Some of the new functions perform uppercase/lowercase conversions and return the larger or smaller of two strings or numbers.

The real novelty is in the way Better BASIC allows user-definable data-typed functions and procedures. The implementation uses the concept of separate workspaces to store the main program and each user-defined function and procedure. Immediately, one realizes the variables and line numbers are local.

Better BASIC handles passing arguments with style. Function and procedure arguments can be passed by value, by reference, or as shared data. In the first case, the passed argument becomes a local constant in the subprograms. It cannot be assigned a new value. However, it can initialize a local variable of the same type, and the variable can have its content altered during the execution of the subprogram.

Better BASIC even allows arguments to have default values. Moreover, subprograms can have enumerated argument lists. Procedure and function overloading, or Procedure Families as the Better BASIC manual calls it, is also implemented. This gives the effect of a procedure capable of accepting a different number and/or type of arguments.

To write such subprograms, you start with the "root" function/procedure, which is written to handle one type of data. All the other overloading subprograms are written such that they have the same root name plus an extended name in the form of .A, .B and so on. These subprograms are free to accept any type of data and perform any manipulation required.

Finally, Better BASIC allows the creation of modules containing *PUBLIC* functions and procedures. Module names can be included in the configuration file to enable their automatic inclusion upon entering Better BASIC.

Better BASIC offers sequential and random access file I/O. No numeric data packing functions are needed. Structured records can be involved in random access I/O. The *SIZE* function is used to return the number of the byte length of a variable.

Graphics allow the programmer to create several windows and define the scales on each. Color selection, plotting points, and the ability to draw lines and shapes and fill areas with colors are some of the features offered. Screens can be saved and recalled from disks, and portions of a screen can be stored in an array.

Error handling is carried out with the *ON ERROR* statement followed by other statements such as *GOTO*, *GOSUB*, procedure calls, or *PRINT*. This enables versatile error handling.

Better BASIC did very well in the speed benchmark test. It even sorted 1,000 integers faster than True BASIC. However, its trig function evaluation speed lags behind. We did not receive the 8087 math support module for the review. One can imagine a boost in speed using it.

### TransEra TBASIC

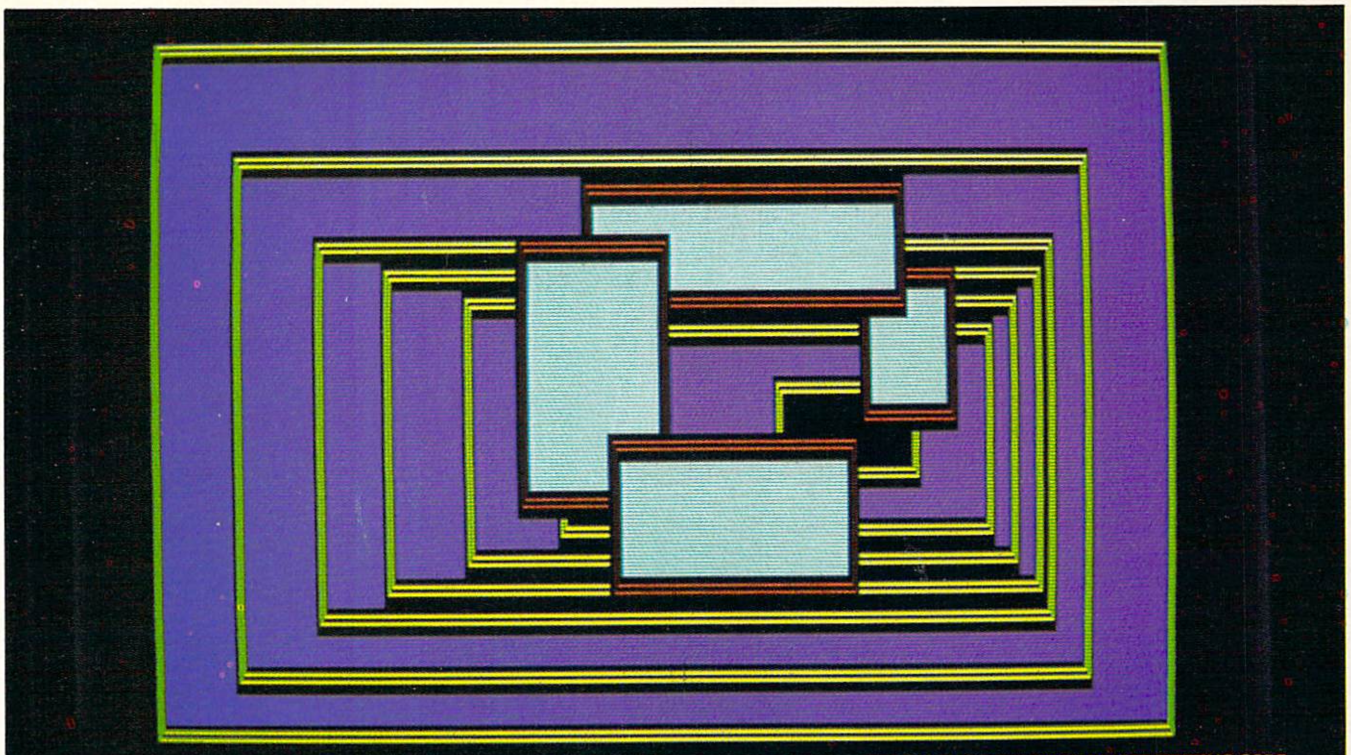
This implementation from TransEra Corp. is semicompiled and has a syntax close to the proposed ANSI BASIC while still maintaining high compatibility with the Tektronic 4050A BASIC. A line editor is used to alter program lines that must start with a line number. TBASIC allows all keywords to be abbreviated to three letters. While this decreases the amount of typing done, program readability will suffer.

TBASIC data types are integers, floating point reals, and strings. No special symbols, like %, are tagged to a variable's name to indicate that it is an integer. Instead, it must be explicitly declared as such.

Dimensioning strings involves declaring the number of strings in an array as well as the maximum string length of each array member. The default string length is 132 characters and the maximum limit is 32K bytes.

TBASIC provides a function to sort single- and multidimensioned string arrays in either ascending or descending order. In the case of string matrices, you can select the column number whose values will be used to arrange the data.

TBASIC offers few, but important, matrix manipulation statements, such as matrix inversion and transposing, calculating the determinant of a matrix, and creating identity matrices. In addition, row and column sums of arrays are provided.



WHEN YOU BUILD A HOUSE . . . YOU DON'T NEED TO MAKE THE WINDOWS YOURSELF. NOW . . . THE SAME IS TRUE WHEN YOU'RE WRITING CODE.

# Why did Simon & Schuster, 3Com, Tymshare, and Revlon choose VSI–The Window Machine?

## (and how come you can buy it for such a low price?)

### Windows With A View Toward The Future

The Window Machine™ occupies only 12K! Written in tight, fast Assembler, it performs like a racing engine...with more power than you'll probably ever need. Yet, it's an engine designed to fit in the vehicle of your choice...from a "stripped-down" 128K IBM PC to a fully loaded AT. The programs you write today will run on the broadest range of machines possible... now, and in the future.

### Windows Bigger Than Your Screen?

Here's where the VSI part of our name fits in. VSI means Virtual Screen Interface. Behind each window, there's a much bigger picture. VSI defines virtual screens rather than just windows. The window itself shows whatever portion of its virtual screen you wish to exhibit at any given point in your program. Each screen can be up to 128 x 255 (columns x rows, or rows x columns). And there are more than 100 screen primitives at your command.

*These are coders' windows... designed to be built into the programs you are writing. They can overlap, move anywhere on the screen, grow, shrink, vanish or blink. They can be bordered in anything from a simple line to flashing asterisks...or even no border at all. And you can have up to 255 of them at a time! Color or monochrome ...of course!*

### Multilingual Windows

You can order The Window Machine with the language interface of your choice: C, Pascal, Compiled Basic, Fortran, Cobol, or PL1. We've even recently completed an interface for Turbo Pascal*, so that now true, full-featured windowing can be utilized with this fine compiler. (Turbo's own built-in "windowing" procedure is extremely limited).

### Windows That Won't Break You

We decided to save you a lot of money. So, we left behind fancy binders, monogrammed slip cases and plastic presentation boxes. Instead, you'll find an extremely powerful tool and a 200 page manual written with an eye toward simplicity, clarity and completeness. (We figured if you wanted ribbons and bows you could always add them yourself.)

And by offering you the product ourselves, we were able to cut out all the middlemen and save you a tremendous amount of money.

**VSI** THE WINDOW MACHINE

*Available for the IBM PC, XT, AT, IBM Compatibles, Wang, T.I., and HP 150*

**The Window Machine Includes:**

**$59.95**

- *Zoom Windows*
- *Multiple Virtual Screens (up to 255)*
- *Choice of Borders (including flashing borders)*
- *Support for all Color and Monochrome Video Attributes (no graphics card required)*
- *Built-in Diagnostics*
- *And much, much more*

**ORDER YOUR COPY OF VSI—THE WINDOW MACHINE TODAY**

**For Visa & MasterCard orders call toll free:**

**800-538-8157 Ext. 824 In CA 800-672-3470 Ext. 824**

**Call Mon.-Fri. 6A.M. to 12P.M., Sat. & Sun. 6A.M. to 8P.M. (P.S.T.)**

**The Window Machine*** $59.95 + $5 Shipping and Handling

LANGUAGE INTERFACE:

☐ Lattice C ☐ Realia Cobol ☐ Microsoft Basic Compiler ☐ Microsoft Fortran
☐ PL1 ☐ Microsoft Pascal ☐ Turbo Pascal (full featured true windowing)

COMPUTER ____________

Name ____________

Address ____________

City ____________ State ____ Zip Code ____________

☐ Check ☐ Money Order ☐ VISA ☐ MasterCard

Card # ____________ Exp. Date ____________

*California residents: tax included. Orders outside USA: Please add $10 for shipping and handling

**30 day Money Back Guarantee**

CL

AMBER
AMBER SYSTEMS
1171 S. Saratoga-Sunnyvale Road
San Jose, CA 95129

**AMBER SYSTEMS, INC. 1171 S. Saratoga-Sunnyvale Road, San Jose CA 95129**

**FOR DEALER INQUIRIES: CALL OUR 800 NUMBER**

**CIRCLE 2 ON READER SERVICE CARD**

---

*Turbo Pascal is a Trademark of Borland International

## General information

| Manufacturer and product | Version tested | Price | Type | Editing | Debugging | Compatibility with MS-BASIC | Ran standard benchmark programs |
|---|---|---|---|---|---|---|---|
| **MS-DOS interpreters** | | | | | | | |
| American Planning Corp. MEGABASIC | 5.01 | $375 | SC | LE | PL, V | fairly compatible | few modifications |
| Control-C BASIC Interpreter | 1.5 | $400 | TI | LE | PL | not compatible | major modifications |
| IBM/Microsoft BASICA | 2.0 | — | TI | SE, LE | PL | superset | no modifications |
| Microsoft MS-BASIC | 5.28 | $350 | TI | LE | PL | — | no modifications |
| Morgan Computing Professional BASIC | 2.01 | $99 | SC | LE | W | subset | no modifications |
| Ryan-McFarland RM-BASIC | 1.0 | $600 | SC | LE | PL | fairly compatible | few modifications |
| Southwest Data Pluto BASIC | 2E.1 | $595 | SC | LE | L, V | not compatible | ran test 1 with major modifications |
| Summit Software Better BASIC | 1.1 | $199 | SC | SE | none | moderately compatible | few modifications |
| TransERA TBASIC | 1.0 | $595 | SC | LE | PL, V | not compatible | few modifications |
| True BASIC | 0.9 | $149.90 | SC | SE, FR | PL, V | not compatible | few modifications |
| WATCOM BASIC | 2.1 | $250 | TI | LE, SE, FR | PL, V | fairly compatible | few modifications |
| **MS-DOS compilers** | | | | | | | |
| Digital Research CBASIC | 2.1 | $600 | NC | — | V | fairly compatible | no modifications |
| IBM/Microsoft IBM BASIC Compiler | 1.0 | — | NC | — | PL | very compatible | no modifications |
| Microsoft Business BASIC Compiler | 1.0 | $450 | NC | — | PL | very compatible | no modifications |
| Microsoft BASIC Compiler | 5.38 | $395 | NC | — | PL | fairly compatible | no modifications |
| MicroWay 87BASIC | 3.03 | $150 | NC | — | PL | very compatible | no modifications |
| Softaid MTBASIC | 2.5 | $49.95 | NC | LE | PL, V | barely compatible | few modifications |
| Sparry BASIC B | 1.2 | $159 | NC | — | PL, V | barely compatible | few modifications |
| **CP/M 86 interpreter** | | | | | | | |
| Digital Research Personal BASIC | 1.2 | $150 | TI | LE | PL, V | very compatible | no modifications |
| **CP/M interpreters** | | | | | | | |
| DeltaSoft DeltaBASIC | 3.0 | $275 | TI | LE | none | fairly compatible | few modifications |
| Micro Mike's BaZic | 08/04 | $150 | TI | LE | none | barely compatible | few modifications |
| Spectrum Logic DBASIC | 3.0 | $475 | TI | LE | PL | very compatible | no modifications |
| **CP/M compilers** | | | | | | | |
| Alcor Systems Multi-BASIC | 1.0 | $89.95 | NC, PC, CA | SE, FR | PL | very compatible | no modifications |
| System/z BASIC/Z | 1.11 | $345 | NC, CA | LE, FR | PL, V | barely compatible | few modifications |
| **Apple Macintosh** | | | | | | | |
| Apple MacBASIC | 0.974 | N.A. | SC | SE, FR | PL, V, W | fairly compatible | few modifications |
| Microsoft MS-BASIC 1.0 | 1.0 | $150 | TI | LE | PL | superset | no modifications |
| Microsoft MS-BASIC 2.0b | 2.0b | $150 | TI | SE, FR | W | superset | no modifications |
| Microsoft MS-BASIC 2.0d | 2.0d | $150 | TI | SE, FR | W | superset | no modifications |

TI = True interpreter.
SC = Semicompiled.
NC = Compiled into native code.
PC = Compiled into p-code.
CA = Compiled into assembly language source.
LE = Line editor.
SE = Screen editor.
FR = Find replace.
PL = Program lines.
V = Variables.
W = Uses windows.

Table 1.

TBASIC offers nine binary operators for bitwise arithmetic and manipulation. This includes shifting, logical shifting, *NOT*, *OR*, *XOR*, bitwise circular rotation, bit setting, and testing. This enables low-level data manipulation used in conjunction with other powerful TBASIC features.

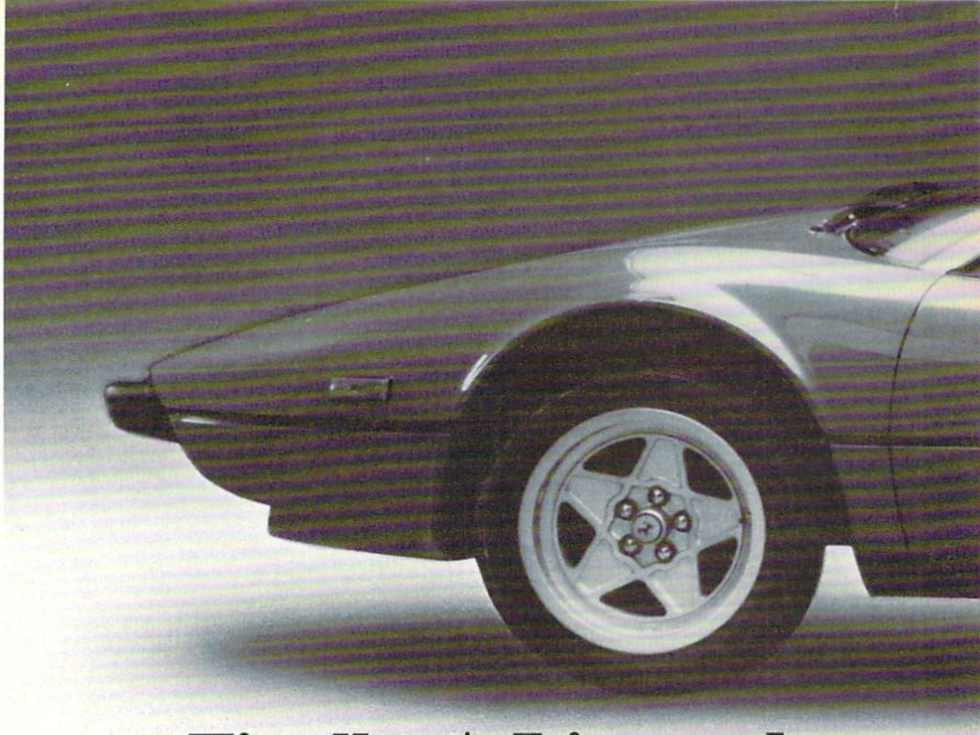
Loop constructs are offered abundantly, beginning with the traditional *FOR-NEXT* loop. *DO-LOOP UNTIL < condition >* will repeat a loop until the tested condition is satisfied. *DO WHILE < condition > -LOOP* offers the equivalent of Microsoft BASIC's *WHILE-WEND* loop. *DO-LOOP* offers an open loop construct. All these loops can be exited by using the *EXIT* statement. The *EXIT* statement may have an optional line number or label to resume program execution in a line other than that past the end of the loop.

With decision making, TBASIC again offers impressive constructs. *IF-THEN-ELSE* can be spread over many lines. The *END IF* statement is used to indicate the end of the *IF* construct. The *SELECT CASE* statement offers another tool for decision making. The tests allowed in each *CASE* clause involve testing if the variable's content matches with a list of values or a range of values. In addition, other logical tests, like greater than or lesser than, can be incorporated. True BASIC is the only other implementation that offers this feature.

While TBASIC restricts user-definable functions to one line, it allows subroutines to occupy multiple lines and declare local variables. Moreover, subroutines can be loaded from external, compiled library files. This offers a versatile method of developing general purpose subroutines or overlays. Once a subroutine is no longer needed, it can be deleted to free the memory space it occupies. TBASIC offers the *PICTURE* subroutine, specialized in drawing shapes quickly. The only other implementation supporting this is True BASIC.

When it comes to graphics, TBASIC means business. In this review, it is rated number one for its graphics among the other BASICs running under the MS-DOS environment. Describing its capabilities and features deserves a separate review. About 54 graphics related statements exist. The user can define the boundary of the viewing surface (*SET VIEWPORT*) and assign the real-world scales (*SET WINDOW*). Plotting points, drawing lines and axes, areas, arcs, circles, and mixing text with graphics are some of the many features available. Shapes can be transformed, rotated, and sheared, enabling graphical animation and simulations.

TBASIC offers sequential and random access records. The *OPEN* statement specifies the type of access. This can be



Finally, A Lint and Make for MS™ - DOS

Get the full range of features C programmers working in UNIX™ have come to expect from their Lint and Make utilities. With Pre-C™ you can detect structural errors in C programs five times faster than you can with a debugger. Find usage errors almost impossible to detect with a compiler. Cross-check multiple source files and parameters passed to functions. Uncover interface bugs that are difficult to isolate. All in a single pass. Capabilities no C compiler, with or without program analyzing utilities, can offer. Pre-C outlints Lint, since you can handle analyses incrementally.

Pre-C's flexible library approach lets you maintain continuity across all programs in your shop, whether you use Pre-C's pre-built libraries, functions you already have, or some you might want to buy.

Plus, you're not limited to one particular library. Pre-C keeps track of all the libraries you're using to make sure that code correctly calls them.

With Pmaker™ you can update and track every module in your program. When you make a change in any source or include file, all you do is run Pmaker. It will recompile changed modules and relink your program. With any compiler or linker you choose. Pmaker can update an object module library when one or several of the object modules are changed. You can use Pmaker to handle any task when a change requires several steps.

Pmaker
Pre-C
Phoenix
Phoenix

Pre-C by Phoenix. $395. Pmaker by Phoenix. $195.

Call (1) 800-344-7200.
In Massachusetts (617) 762-5030.

Or, write: Phoenix Computer Products, Corp., 1420 Providence Highway, Suite 115, Norwood, MA 02062.

Phoenix

PROGRAMMERS' PFANTASIES BY PHOENIX

Pre-C and Pmaker are trademarks of Phoenix Computer Products Corporation.
MS-DOS is a trademark of Microsoft Corporation. UNIX is a trademark of Bell Laboratories.

CIRCLE 24 ON READER SERVICE CARD

## Data types and variables

| Product and manufacturer | Binary constants | Octal constants | Hexadecimal constants | Long integer | Unsigned integers | Double precision reals | BCD math | 8087 support | Support record structure (Pascal style) | Max string length (bytes) | Max variable name length (characters) | Array lower bound > 1 | Matrix operations | Matrix I/O |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **MS-DOS interpreters** | | | | | | | | | | | | | | |
| American Planning Corp. MEGABASIC | yes | yes | yes | yes | no | yes | yes | no | no | 65K | 250 | no | no | no |
| Control-C BASIC Interpreter | no | no | yes | no | no | no | no | no | no | AM | 2 | no | no | no |
| IBM/Microsoft BASICA | no | yes | yes | no | no | yes | no | no | no | 255 | 40 | no | no | no |
| Microsoft MS-BASIC 5.28 | no | yes | yes | no | no | yes | no | no | no | 255 | 40 | no | no | no |
| Morgan Computing Professional BASIC | no | yes | yes | yes | yes | yes | yes | yes | no | 512 | 40 | yes | no | no |
| Ryan-McFarland RM-BASIC | no | no | no | yes | no | yes | no | no | no | 255 | 40 | no | yes | no |
| Southwest Data Pluto BASIC | no | no | yes | no | no | yes | no | no | no | AM | 2 | no | no | no |
| Summit Software Better BASIC | yes | yes | yes | no | no | yes | yes | yes | yes | 32K | — | no | no | no |
| TransERA TBASIC | no | no | no | no | no | yes | no | yes | no | 32K | 31 | no | yes | no |
| True BASIC | no | no | no | no | no | yes | no | yes | no | 32K | 31 | yes | yes | yes |
| WATCOM BASIC | no | no | no | no | no | yes | no | yes | no | 32K | 31 | no | yes | no |
| **MS-DOS compilers** | | | | | | | | | | | | | | |
| Digital Research CBASIC | yes | no | yes | no | no | no | no | no | no | 32K | 31 | no | no | no |
| IBM/Microsoft IBM BASIC Compiler | no | yes | yes | no | no | yes | no | no | no | 32K | 40 | no | no | no |
| Microsoft Business BASIC Compiler | no | yes | yes | no | no | yes | yes | yes | no | 32K | 40 | no | no | no |
| Microsoft BASIC Compiler | no | yes | yes | no | no | yes | no | no | no | 32K | 40 | no | no | no |
| MicroWay 87BASIC | no | yes | yes | no | no | yes | no | no | no | 32K | 40 | no | no | no |
| Softaid MTBASIC | no | no | yes | no | no | no | no | no | no | 127 | 7 | no[1] | no | no |
| Sparry BASIC B | no | no | no | no | no | no | no | no | no | AM | 40 | no | no | no |
| **CP/M 86 interpreter** | | | | | | | | | | | | | | |
| Digital Research Personal BASIC | yes | yes | yes | no | no | yes | no | no | no | 255 | 31 | no | no | no |
| **CP/M interpreters** | | | | | | | | | | | | | | |
| DeltaSoft DeltaBASIC | no | no | no | no | no | yes | no | no | no | 255 | 8 | no | no | no |
| Micro Mike's BaZic | no | no | no | no | no | yes | no | no | no | AM | 2 | no | no | no |
| Spectrum Logic DBASIC | no | yes | yes | no | no | yes | no | no | no | 255 | AM | no | no | no |
| **CP/M compilers** | | | | | | | | | | | | | | |
| Alcor Systems Multi-BASIC | yes | yes | yes | no | no | yes | no | no | no | AM | 255 | no | no | no |
| System/z BASIC/Z | yes | no | yes | yes | yes | yes | yes | no | no | 255 | 250 | no | no | no |
| **Apple Macintosh** | | | | | | | | | | | | | | |
| Apple MacBASIC | no | no | no | yes | no | yes | no | no | no | 65K | AM | no | no | no |
| Microsoft MS-BASIC 1.0 | no | yes | yes | no | yes | yes | no | no | no | 32K | 40 | no | no | no |
| Microsoft MS-BASIC 2.0b | no | yes | yes | no | yes | no | yes | no | no | 32K | 40 | no | no | no |
| Microsoft MS-BASIC 2.0d | no | yes | yes | no | yes | yes | no | no | no | 32K | 40 | no | no | no |

AM = Available Memory.

Table 2.

# Creators of COMPUTER LANGUAGE Sponsor the C Expert Forum

C Seminar/Workshop

September 16-18
Cambridge, Mass

Never before have so many leaders in the C programming field gathered for one event. The C Seminar/Workshop will be an exciting forum on the latest technical innovations and C language developments. Best of all, you'll experience a practical, hands-on approach in small workshop sessions. All this in the beautiful autumn foliage of New England, just four blocks from Harvard Yard. The C Seminar/Workshop is brought to you by the publishers of COMPUTER LANGUAGE.

The cost for this comprehensive 2½ day event is only $695. Sign up by June 30th and receive a $100 early bird discount.

## CURRICULUM

**Speakers**

**Jim Brodie,** *ANSI C committee chairman:* Overview of the ANSI Standardization Effort
**P.J. Plauger,** *author, ANSI C committee secretary:* Programming Style and C
**Larry Rosler,** *ANSI C language chairman:* Language Standardization Issues
**Tom Plum,** *author:* Efficiency of C Programs
**Heinz Lycklama,** */usr/group UNIX chairman:* UNIX Perspective on C
**Leor Zolman,** *compiler writer:* Porting C Programs between Operating Systems
**Robert Ward,** *C User's Group coordinator:* Structured Methods of Debugging C

**Workshops**

*(Subject to change based on availability)*

**Seminar participants will be able to choose four from this list:**

- Debugging Techniques
- Interpreters in a Development Environment
- Programming for Portability
- Efficient Code Generation
- Cross Compilers
- Network Data Base Theory and C
- Object-File Formats for UNIX Systems
- Philosophy and Methodology of Benchmarks
- ANSI Standards: Questions & Answers
- Code Readability and Organization
- Asynchronous Communications
- Writing Extensions to C
- C / UNIX System Subroutine Interfaces
- Porting C between CP/M, MS-DOS, and UNIX

## C Seminar/Workshop Registration Form

**Please enroll me in the C Seminar:**

☐ Early Bird $595 (pay by 6/30/85)
☐ Single $695
☐ Multiple
(3 or more enrollments get $100 discount)
☐ I do not wish to enroll at this time but please send me more information.

**Method of Payment:**

☐ Check Enclosed
☐ Bill My Company

Make check payable to:
C.L. Publications Inc.

Name & title ____________________
Name & title ____________________
Name & title ____________________
Company ____________________
Address ____________________
City, State, Zip ____________________
Phone ____________________

CA85

**COMPUTER LANGUAGE Seminar**
**131 Townsend St.**
**San Francisco, Calif. 94107**
**(415) 957-9353**

C

**Software Development**

**PCDOS/MSDOS**

**Complete C Compiler**

- Full C per K&R
- Inline 8087 or Assembler Floating Point, Auto Select of 8087
- Full 1Mb Addressing for Code or Data
- Transcendental Functions
- ROMable Code
- Register Variables
- Supports Inline Assembler Code

**MSDOS 1.1/2.0 Library Support**

- All functions from K&R
- All DOS 2.0 Functions
- Auto Select of 1.1 or 2.0
- Program Chaining Using Exec
- Environment Available to Main

**c-window™ Symbolic Debugger**

- Source Code Display
- Variable Display & Alteration Using C Expressions
- Automatic Commands
- Multiple Breakpoints by Function & Line Number

**8088/8086 Assembler**

- FAST — Up to 4 times Faster than IBM Assembler
- Standard Intel Mnemonics
- Compatible with MSDOS Linker
- Supports Full Memory Model

**8088 Software Development Package** $199.00

Includes: C Compiler/Library, c-window, and Assembler, plus Source Code for c-systems Print Utility

**c-systems**

P.O. Box 3253
Fullerton, CA 92634
714-637-5362

CIRCLE 16 ON READER SERVICE CARD

one of the following modes: create, read-only, write-only, read-and-write, update and append.

Some modes are restricted to dealing with new or old files. For example, only old files can be *READ*. Attempting to *READ* a new file will result in error. TBASIC provides functions to pack and unpack numeric arrays into strings. This can be used to control array storage on file. In addition to file I/O, TBASIC offers the versatile general purpose interface bus (GPIB) I/O.

Errors are handled by the *ON ERROR* statement. A single or a range of error codes can be specified to tackle peculiar and anticipated errors. A range of data file buffer numbers can also be included. The action taken is either jumping to a specific program line or calling a subroutine.

TBASIC provides functions to report the error code number, location, and file number (in case of file I/O error). User-definable error code is also allowed. They help in creating error conditions for cases like the occurrence of fatal logical errors in run time.

TBASIC provides the user with 56 statements that either inquire about or set different operating parameters, such as trigonometric angle mode, option base, window and viewport dimensions, and many more.

TBASIC ran the benchmark faster than MS-BASIC but much slower than True BASIC. The lag is less for the trigonometric test.

### True BASIC

Behind this implementation from True BASIC Inc. are John Kemeny and Thomas Kurtz, the founders of the BASIC language. True BASIC, which is more structured, conforms to the new proposed ANSI BASIC.

Programs written in True BASIC need no line numbers. However, once a *GOTO*, *GOSUB*, *THEN* or *ELSE* statement followed by a line number is introduced, all the program lines must be numbered. The screen is split into two areas: program editing and command windows. The keyboard function keys are used to go back and forth between the two. In addition to the screen editor, some commands can perform a variety of tasks, such as including another program file, moving and copying part of a file, searching and/or replacing words or phrases.

True BASIC recognizes two data types only: numbers and strings. No explicit distinction between integers and floating point variables exists. Matrix manipulations are available, including matrix I/O, mathematical operation with scalar and other matrices, determinant, matrix inversion, and transposition.

Strings can be as long as 32K bytes. In handling portions of a string, the first and last positions are indicated. This differs from Microsoft BASIC where the first position and a character count are used.

Loops are implemented in a variety that pleases the heart. The open loop *DO-LOOP*, *DO WHILE <cond>-LOOP*, *DO-LOOP UNTIL <cond>* and the double test loop *DO WHILE <cond1> -LOOP WHILE <cond2>* are available. This exceeds even the loops in Pascal. The *EXIT-DO* statement is used to exit loops and is especially useful for the open loops.

Decision making includes multiline *IF-THEN-ELSE*, *IF-THEN-ELSEIF*, and even the *SELECT CASE*, which closely resembles the *CASE* statement in Pascal, Modula-2, and Ada. Each *CASE* clause allows comparison and expression to a list of values, a range of values, or performs a logical test. This is more powerful than what Pascal and Modula-2 offer. All the loop and decision-making constructs indicate an invitation to more stuctured code that avoids using *GOTO*s.

True BASIC provides many numeric and string functions. They include a good number of new (relative to Microsoft and Applesoft BASIC) functions, such as round, truncate, and calculate the modulo and remainder of numbers. New string functions include lowercase/uppercase conversion and trimming of leading or trailing spaces. User-definable functions are also supported.

Mutliline functions are allowed. They can be internal or external. Internal functions must be defined before the *END* statement. External functions are defined either in separate library files or past the *END* statement. True BASIC demands that every program ends with a unique *END* statement. This contrasts with other popular implementations where programs can have many *END*s scattered throughout the code.

Functions may take simple variables, arrays, and file I/O buffer numbers as arguments. All variables not passed in the argument list are local. True BASIC allows named subroutines that are *CALL*ed. Like functions, they can be internal or external, accept arguments, and have local variables. One difference between functions and subroutines is that the arguments in the former are passed by value (that is, a copy is made), while in the latter they are passed by reference. Thus only subroutines can alter the value of their arguments.

Graphics are implemented and are capable of plotting points, drawing lines, ellipses, and areas, and mixing text with graphics. The user can define his or her own real-world coordinates. A special set of *BOX* statements allow fast drawing to accommodate animated display. The *GET MOUSE* statement permits use of a mouse.

## WIZARD C

**Fast compiles, fast code and great diagnostics make Wizard C unbeatable on MSDOS. Discover the powers of Wizard C:**

- ALL UNIX SYSTEM III LANGUAGE FEATURES.
- UP TO A MEGABYTE OF CODE OR DATA.
- SUPPORT FOR 8087 AND 80186.
- FULL LIBRARY SOURCE CODE, OVER 200 FUNCTIONS.
- CROSS-FILE CHECKS OF PARAMETER PASSING.
- USES MSDOS LINK OR PLINK-86.
- CAN CALL OR BE CALLED BY PASCAL ROUTINES.
- IN-LINE ASSEMBLY LANGUAGE.
- 240 PAGE MANUAL WITH INDEX.
- NO LICENSE FEE FOR COMPILED PROGRAMS.

**The new standard for C Compilers on MSDOS!**

**Only $450**

**WSS**

For more information call (617) 641-2379
Wizard Systems Software, Inc.
11 Willow Ct., Arlington, MA 02174
Visa/Mastercard accepted

**CIRCLE 86 ON READER SERVICE CARD**

## Poor Person Software

Introduces

# Write-Hand-Man

Desk accessories for CP/M

**Write-Hand-Man** lets you take notes, check phone numbers, make appointments, and countless other tasks without leaving Wordstar, dBase, Multiplan, or any other application. Enter **Write-Hand-Man** with a single keystroke and choose the program you want. When you leave **Write-Hand-Man,** your application continues normally.

**$49.95** plus tax delivers **Write-Hand-Man** and 4 companion programs; **Notepad, Phonebook, Calendar,** and **Termcomm.** User written programs are easily added. All you need is M80 or some other LINK-80 compatible assembler.

Other CP/M products available from **Poor Person Software: Poor Person's Spooler** ($49.95), **Poor Person's Spelling Checker** ($29.95), **Poor Person's Spread Sheet** ($29.95), **Keyed Sequential Files** ($39.95), **Poor Person's Menus** ($29.95), **aMAZEing Game** ($29.95), **Window System** ($29.95), **Crossword Game** ($39.95), **Mailing Label Processor** ($29.95). Shipping included.

All products available on IBM 8 inch and Northstar 5 inch disks. Other 5 inch formats add $5 handling charge. No credit cards.

**Poor Person Software**
3721 Starr King Circle
Palo Alto, CA 94306
tel 415-493-3735

CP/M is a registered trademark of Digital Research

**CIRCLE 67 ON READER SERVICE CARD**

# SuperSoft Diagnostics

## When Reliability Counts

Protect yourself from time-robbing system failure. Pinpoint costly hardware problems before they cause serious trouble. Diagnostics II from SuperSoft can help you eliminate hardware problems, service calls, and data loss due to system failure.

### End Users

Diagnostics II is the finest set of system diagnostics available for microcomputers. It thoroughly checks memory, CPU, terminal, printer, and disk drives – isolating many problems to the chip level. It checks both standard and non-standard components, including non-IBM add-ons. The memory test is particularly powerful; incorporating a quick test, walking bit test, burn-in test, and speed test to make sure every bit of memory is completely reliable.

### Manufacturers

Hardware manufacturers, systems houses, and service organizations – we can tailor our diagnostics software to your specific needs. We have developed custom diagnostics for companies such as NCR, XEROX, MORROW DESIGNS, and SONY. From easy to operate user level diagnostics to exhaustive service level tests, we can provide the expertise you need.

So whether you're an end user, service technician, or system manufacturer, get SuperSoft's Diagnostics II for yourself and keep your system in great shape.

Diagnostics II
(for all PC DOS, MS DOS, CP/M-86, and CP/M-80 systems): $125
Call for pricing on customized versions.

TO ORDER CALL
800-762-6629
(in Illinois call 217-359-2112)

or SEND YOUR CHECK OR CREDIT CARD INFORMATION TO THE ADDRESS BELOW. Add $3 shipping U.S., $6 Canada, $20 all other areas. Please specify your computer and operating system. (C.O.D. orders also accepted)

**SuperSoft**

SuperSoft, Inc. P.O. Box 1628,
Champaign, IL 61820
Telex: 270365 SUP ACI CHM

**CIRCLE 77 ON READER SERVICE CARD**

## Program flow control, decision making, and error handling

| Manufacturer and product | Must use line number | Alpha-numeric labels | Only single line IF-THEN-ELSE | WHILE loop | REPEAT loop | Open loop | Use EXIT loop | CASE selection |
|---|---|---|---|---|---|---|---|---|
| **MS-DOS interpreters** | | | | | | | | |
| American Planning Corp. MEGABASIC | yes | yes | yes | yes | no | yes | yes | no |
| Control-C BASIC Interpreter | yes | no | yes[1] | no | no | no | no | no |
| IBM/Microsoft BASICA | yes | no | yes | yes | no | no | no | no |
| Microsoft MS-BASIC 5.28 | yes | no | yes | yes | no | no | no | no |
| Morgan Computing Professional BASIC | yes | yes | yes | yes | no | no | yes | no |
| Ryan-McFarland RM-BASIC | yes | yes | yes | no | no | no | no | no |
| Southwest Data Pluto BASIC | yes | no | no | no | no | no | yes | no |
| Summit Software Better BASIC | yes[2] | no | yes | yes | yes | yes | yes | no |
| TransERA TBASIC | yes | yes | no | yes | yes | yes | yes | yes |
| True BASIC | no | no | no | yes | yes | yes | yes | yes |
| WATCOM BASIC | yes | no | no | yes | no | no | no | no |
| **MS-DOS compilers** | | | | | | | | |
| Digital Research CBASIC | no | yes | no | yes | no | no | no | no |
| IBM/Microsoft IBM BASIC Compiler | no | no | yes | yes | no | no | no | no |
| Microsoft Business BASIC Compiler | no | yes | yes | yes | no | no | no | no |
| Microsoft BASIC Compiler | no | no | yes | yes | no | no | no | no |
| MicroWay 87BASIC | no | no | yes | yes | no | no | no | no |
| Softaid MTBASIC | yes | no | yes | no | no | yes | yes | no |
| Sparry BASIC B | no | no | yes | no | no | no | no | no |
| **CPM/86 compiler** | | | | | | | | |
| Digital Research Personal BASIC | yes | no | no | yes | no | no | no | no |
| **CP/M interpreters** | | | | | | | | |
| DeltaSoft DeltaBASIC | yes | no | yes | no | no | no | no | no |
| Micro Mike's BaZic | yes | no | yes | no | no | yes | yes | no |
| Spectrum Logic DBASIC | yes | no | yes | no | no | no | no | no |
| **CP/M compilers** | | | | | | | | |
| Alcor Systems Multi-BASIC | no | yes | no | yes | no | no | no | no |
| System/z BASIC/Z | yes | yes | yes | yes | yes | yes | no | no |
| **Apple Macintosh** | | | | | | | | |
| Apple MacBASIC | no | yes | no | no | no | yes | yes | yes |
| Microsoft MS-BASIC 1.0 | yes | no | no | yes | no | no | no | no |
| Microsoft MS-BASIC 2.0b | no | yes | no | yes | no | no | no | no |
| Microsoft MS-BASIC 2.0d | no | yes | no | yes | no | no | no | no |

1. No THEN clause. 2. Except declaration section. 3. Only IF-THEN.

Table 3.

## YOUR CODE MAY BE WASTING ITS TIME!
## THE PROFILER™ CAN HELP . . .

- **Statistical Execution Profiler**
- **Works with any language**
- **Completely configurable**
- **Up to 16 partitions in RAM/ROM**
- **Time critical code optimization**
- **Abnormal code behavior tracking**
- **Graphic presentation of results**
- **Easy to use menu interface**

**THE PROFILER** is a software package which gives you, the programmer, a powerful tool for locating time consuming functions in your code and allows you to performance tune your program. With **THE PROFILER** you can determine where to optimize your code for maximum benefit, then measure the results of your efforts.

Using **THE PROFILER,** you can answer questions like:
Where is my program spending its time?
Why is my program so slow? What is it doing?
Is my progam I/O bound? CPU bound? Are data buffers large enough?
How much improvement did my changes make?

**THE PROFILER** is completely software based and consists of a system resident driver and a monitor program. The memory partitions can range from 1 byte to 1 megabyte in size and can be anywhere in the address space.

**NO ADDITIONAL HARDWARE IS REQUIRED!**

Requires an IBM PC or compatible system with a minimum 64k and one drive.

**THE PROFILER** is available for $125.00 **(New Low Price)** from DWB Associates or ask your software dealer. To order or for more information, call or write DWB Associates. VISA/MC accepted. Dealers welcome.

IBM is a trademark of IBM Corp. MSDOS is a trademark of Microsoft Corp.
THE PROFILER is a trademark of DWB Associates.

**dwb**
**Associates**

**P.O. Box 5777**
**Beaverton, OR 97006**
**(503)629-9645**

**CIRCLE 21 ON READER SERVICE CARD**

**DEZIGN**

## THE STRUCTURED PROGRAMMING TOOL FOR MODERN TIMES

- Design your programs right on the screen, using modern techniques based on the popular Jackson Structured Programming method (JSP)!
- **DEZIGN** is more than just another flowcharting tool. It is an integrated tool for designing and documenting programs and for generating ADA, C, PASCAL, and PL/I source code, as well as dBASE II and dBASE III command files.
- **DEZIGN** enables you to create Data and Program Structure Diagrams using the Sequence, Selection (IF-THEN-ELSE), and Iteration (DO WHILE) constructs; assign detailed statements to the diagrams; and synthesize source code from the control logic represented on the diagrams and the detailed statements assigned to them.
- **DEZIGN** lists for $200. It runs on the IBM PC, XT, or AT and requires 128K RAM, one disk drive, and an 80-column color or monochrome display.
  - DEZIGN-PC runs under DOS 2.0, 2.1, and 3.0.
  - DEZIGN-86 runs under CP/M-86 1.1.
- Want to learn more? Please contact us concerning pricing and availability of JSP reference texts and seminars.

**ZEDUCOMP • P.O. BOX 68 • STIRLING, NJ 07980**
**(201) 755-2262**

dBASE II and dBASE III are trademarks of Ashton-Tate, Inc.

**CIRCLE 83 ON READER SERVICE CARD**

## Z80 System Owners Join The 16/32 Bit Revolution Through Evolution

**Starting At**
**$695.00**

### CO-PROCESSING

**The most cost effective way for Z80 system owners to obtain 16/32 bit processing power and software compatibility is via the HSC CO-16 Attached Resource Processor.**

CO-16 is compatible with any Z80 system running CPM 2.2 or CPM 3. A few examples include:
- KAYPRO 2/4/10 • TRS 2/3/12/16
- AMPRO LITTLE BOARD
- HEATH 89 • SUPERBRAIN
- XEROX 820 • TELEVIDEO 802/803
- MORROW • EPSON QX-10
- LOBO • OSBORNE 1/EXEC
- CROMEMCO • Plus many more

### CO-16

Every CO-16 is delivered with • 16/32 bit micro processor • 16 bit Operating System • 256 Kilo RAM • Z80 interface • 16 bit RAM disk driver • CPM80 2.2 RAM disk driver • CPM 2.2 or CPM 3 compatibility • sources with tools • hardware diagrams • board level or case with power supply.

### CO-1686

The only Z80 16 bit co-processor includes • INTEL 8086 • 6Mhz no wait states • MSDOS 2.11 • IBM BIOS emulator • Memory expansion to 768K • 8087 math co-processor • 3-channel Real Time Clock • Runs many IBM PC applications • Shares hard disk space with CPM80 • PC diskette compatilibility on many systems • CPM86 • Concurrent CPM is coming.

### CO-1668

The only Z80 16/32 bit co-processor includes • MOTOROLA 68000 microprocessor • 6 Mhz no wait states • CPM68K • Full "C" compiler with UNIX V7 library and floats • Memory expansion to 1.25 million bytes • NS16081 math co-processor • Real Time Clock • Complete software development environment • 100% file compatible with CPM80 • OS9/68 UNIX look alike coming in February.

Dealer, Distributor and OEM's invited

**Hallock Systems Company, Inc.**
**267 North Main Street**
**Herkimer, N.Y. 13350**
**(315) 866-7125**

HSC INC.

**CIRCLE 12 ON READER SERVICE CARD**

NEW FEATURES

(Free update for our early customers!)

- Edit & Load multiple memory resident files.
- Complete 8087 assembler mnemonics.
- High level 8087 support.
  Full range transcendentals (tan, sin, cos, arctan, logs and exponentials)
  Data type conversion and I/O formatting.
- High level interrupt support.
  Execute Forth words from within machine code primitives.
- 80186 Assembler extensions for Tandy 2000, etc.
- Video/Graphics interface for Data General Desktop Model 10

HS/FORTH

- Fully Optimized & Tested for:
  IBM-PC IBM-XT IBM-JR
  COMPAQ EAGLE-PC-2
  TANDY 2000 CORONA
  LEADING EDGE
  (Identical version runs on almost all MSDOS compatibles!)
- Graphics & Text (including windowed scrolling)
- Music - foreground and background
  includes multi-tasking example
- Includes Forth-79 and Forth-83
- File and/or Screen interfaces
- Segment Management Support
- Full megabyte - programs or data
- Complete Assembler (interactive, easy to use & learn)
- Compare
  BYTE Sieve Benchmark jan 83
  HS/FORTH 47 sec BASIC 2000 sec
  w/AUTO-OPT 9 sec Assembler 5 sec
  other Forths (mostly 64k) 70-140 sec

FASTEST FORTH SYSTEM AVAILABLE.

TWICE AS FAST AS OTHER FULL MEGABYTE FORTHS!

(TEN TIMES FASTER WHEN USING AUTO-OPT!)

HS/FORTH, complete system only: $250.

Visa Mastercard

Add $10. shipping and handling

HARVARD SOFTWORKS

PO BOX 2579
SPRINGFIELD, OH 45501
(513) 390-2087

CIRCLE 47 ON READER SERVICE CARD

True BASIC has implemented special subroutines called *PICTURE*s to draw shapes. They are called by a *DRAW* statement. In addition, five built-in procedures are available to transform shapes by shifting, scaling, rotating, and gearing.

Windows in graphics mode are treated a bit like an output file. Each has a window number, coupled with the actual screen coordinates to define its location. Switching from one window to another is done by using *WINDOW <number>*. Windows are closed, just like files, when no longer needed.

Data files can store information as text, random access records, and bytes. When opening a new file, you can specify the access, creation, and organization modes as well as the record size.

Three data access modes—input-only, output-only and read-and-write—and three creation modes—must-be-new, must-be-old, and new-or-old—are available. The organization mode specifies which of the three data-file types are to be used. Record lengths are used for byte and record data files. True BASIC provides two routines to pack integers into strings and vice versa. This is used in random access records manipulation.

Error handling in True BASIC is indeed powerful. If a program anticipates run-time error in a certain block of statements, they are surrounded by *WHEN ERROR IN* and *USE* keywords, followed by the error handling statements. These statements are terminated by *END WHEN*. This is a very structured and efficient error trapping scheme. Certain functions return the error code number, last error message, and error location. In addition, True BASIC allows the creation of 999 user-defined error codes and messages.

True BASIC performed very well in the benchmarks. This is due to the fact that it is a semicompiler, just like Professional BASIC but faster. Perhaps its nondiscrimination between integers and reals enables it to operate faster. Among interpreters and semicompilers, it is the fastest. Good work, Kemeny and Kurtz!

### WATCOM BASIC

WATCOM Products Inc. is a Canadian company that markets and continues to develop a language that originated at the Univ. of Waterloo in Ontario. It is a very capable language, well suited to the academic surroundings where it was born.

WATCOM BASIC can be obtained for a very large number of machines ranging from small microcomputers (SuperPET) through large mainframes (IBM 370). WATCOM also offers educational material and courses to accompany the language.

The version of WATCOM BASIC we tested came with four loadable modules: editor, matrix package, graphics package, and network package. If you have no need for one or more of these packages, you can load the interpreter without them, saving your RAM for more important code.

Two of the features we liked most are the matrix math capability and the named procedures. WATCOM BASIC has 21 different statements for manipulating matrices. A single statement can add, subtract, or multiply two matrices. Other statements can invert, zero, or initialize individual matrices. This matrix power is very handy for number-crunching applications.

Also useful for number crunching is the integrated 8087 support. WATCOM BASIC will automatically use the 8087 if it is present in your IBM PC. This has a very beneficial effect on math-intensive program execution speed. Just look at the results in the CLTRIG benchmark times.

The named procedures are invaluable for writing modular code, something BASIC needs. The ability to pass parameters when calling a named procedure goes a long way toward making BASIC a production language. It makes the code much more readable.

WATCOM has included some graphic capability with its interpreter. Though not as comprehensive as the graphic abilities of IBM BASICA 3.0, the language extensions do make simple graphing possible, which is better than most other third-party BASIC interpreters.

One problem with the wide diversity of the WATCOM interpreter can be seen in the manuals. A primer and reference manual covers the bulk of the language and appears to have been written to cover all versions of WATCOM BASIC.

The editor has its own manual in an IBM PC version, and a third manual is for IBM PC-related language extensions and explanations of machine-specific implementation questions. Three manuals made for quite a bit of page turning to get answers to operational questions. We were never quite sure whether to use the general reference manual or the IBM PC-specific manual.

## MS-DOS COMPILERS

### Digital Research CBASIC

Digital Research Inc.'s CBASIC is a business compiler version of BASIC. The CBASIC compiler produces fast native code programs. Programs written for CBASIC use no line numbers. Instead, alphanumeric and numeric labels are used with *GOTO*, *GOSUB*, *THEN* and *ELSE* statements. Numeric labels may look like floating point constants (that is, a label 22.5 is legal).

CBASIC has three types of identifiers: integers, reals and strings. Integer con-

## ADVANCED PROCESSORS ARE OK
## TUNED SOFTWARE IS BETTER

Performance. Every user wants more of it.

Advanced hardware is one solution. But advanced hardware is difficult to obtain and expensive. Meanwhile, there are MILLIONS of existing machines waiting for a new generation of faster, more robust software. How will your software measure up?

Performance optimization should be considered part of the software development cycle.

Many programs spend 90% of the time executing less than 10% of the code. A small amount of time invested in optimizing these busy areas can yield big performance dividends. But rarely is it obvious just what pieces of source code account for 90% of the processor time. You need a sieve that can quickly isolate the big pieces (timewise) of your program.

CODE SIFTER is the tool you need to identify the time consuming sections of your EXE or COM file.

It observes your running program and generates a statistical report that indicates which areas of your program are prime candidates for your optimization efforts. The statistical output includes the symbols from your LINK map and can be directed to your display, printer, or to a file.

CODE SIFTER has advanced features.

Like a user friendly interface with function keys and online help. An adjustable sampling rate provides more accurate statistical data. An iteration option can rerun your program a number of times to reduce the busy areas to very narrow ranges. A demonstration program (with source) is provided for use with the step by step tutorial in the manual so that you can become productive in just a few minutes. CODE SIFTER works with any source language and does NOT require a knowledge of assembler to use. And if you are still not convinced – we will loan you a demonstration version.

David Smith Software
Box 25A R.D.#3
Oxford, N.Y. 13830
(607) 843–6209

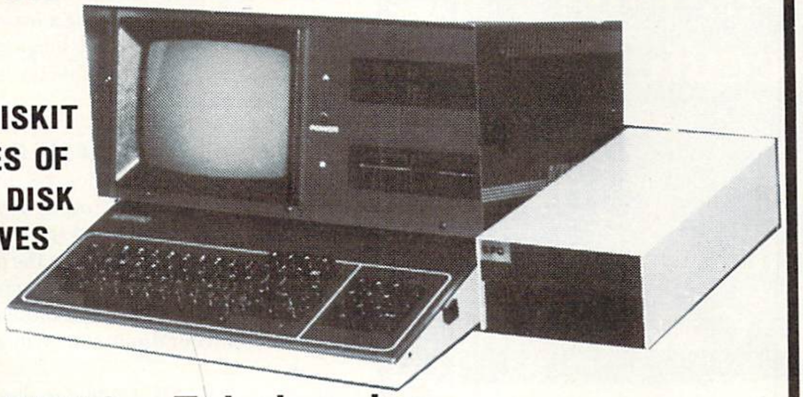
$119

Requires
IBM–PC or XT
128K DOS 2.x

IBM is a registered trademark of International Business Machines Corporation

**CIRCLE 36 ON READER SERVICE CARD**

## FOR THE SERIOUS KAYPRO® USER



**THE DISKIT SERIES OF HARD DISK DRIVES**

**now with**

## ZCPR3 by Echelon, Inc.

Now you can add from 5 to 40 Megabytes of fast-access Winchester storage to your KAYPRO 2, 4, or 10. The DISKIT is only 4 inches high; 5.7 if you get the two drive model with the *removable* 5 or 10 Mb. cartridge, and weighs less than 10 pounds. Easily disconnect DISKIT from the computer whenever you want, and if more capacity is required, just swap your drive for a larger model.

Our DISKIT Model 10 has 10.8 Megabytes of *formatted* capacity . . . **20%** more than a Kaypro 10, and runs about twice as fast. Installs in minutes.

Call SPC now and ask for more information. Quantity and prepayment discounts are available.

**SYSTEMS PERIPHERALS CONSULTANTS**
9747 Business Park Avenue
San Diego, CA 92131
(619) 693-8611

**CIRCLE 91 ON READER SERVICE CARD**

## YOU NEED A GOOD LIBRARY

**C POWER PACKS™**

## COMPLETE SOURCES NO ROYALTIES

**COMPREHENSIVE** C Power Packs include over 1000 functions which provide an integrated environment for developing your applications efficiently. "This is a beautifully documented, incredibly comprehensive set of C Function Libraries."
– Dr. Dobb's Journal, July 1984

**USEFUL** "...can be used as an excellent learning tool for beginning C Programmers..."
– PC User's Group of Colorado, Jan. 1985

**FLEXIBLE** Most Compilers and all Memory Models supported.

**RECOMMENDED** "I have no hesitation in recommending it to any programmer interested in producing more applications code, using more of the PC capabilities, in much less time." – Microsystems, Oct. 1984

| | |
|---|---|
| ■ **PACK 1: Building Blocks I**<br>DOS, Keyboard, File, Printer, Video, Async | $149 |
| ■ **PACK 2: Database**<br>B-Tree, Virtual Memory, Lists, Variable Records | $399 |
| ■ **PACK 3: Communications**<br>Smartmodem™, Xon/Xoff, X-Modem, Modem-7 | $149 |
| ■ **PACK 4: Building Blocks II**<br>Dates, Textwindows, Menus, Data Compression, Graphics | $149 |
| ■ **PACK 5: Mathematics I**<br>Log, Trig, Random, Std Deviation | $99 |
| ■ **PACK 6: Utilities I (EXE files)**<br>Arc, Diff, Replace, Scan, Wipe | $99 |

Master Card/Visa, $7 Shipping, Mass. Sales Tax 5%

**ASK FOR FREE DEMO DISKETTE**

NOVUM ORGANUM INC.

SOFTWARE HORIZONS INC.

165 Bedford St., Burlington, MA 01803
(617) 273-4711

**CIRCLE 25 ON READER SERVICE CARD**

stants may be decimal, hexadecimal or binary. Strings can contain up to 32K bytes. An *LPRINTER* statement directs all the output of a *PRINT* statement to the line printer. The *CONSOLE* statement restores the *PRINT* output to the screen.

Loops in CBASIC consist of *FOR-NEXT* and *WHILE-WEND* loops. Decision making employs multiline *IF-THEN-ELSE*. The backslash symbol is used to indicate continuation of a statement group over several lines. CBASIC has no *ELSEIF* clauses. It can be simulated by nesting the *IF-THEN-ELSE* construct.

CBASIC provides functions similar to those in Microsoft BASIC. User-definable functions are implemented. They are not restricted to one line and can be external. These functions enable separate compilation and the creation of program modules. All formal parameters and variables in a declaration are local.

Arrays declared in multiline functions are global. However, they are recreated every time a call is made, erasing previous values. Functions are invoked either in an expression or with a *CALL* statement (for multiline functions only). The *CALL* allows them to simulate subroutines.

The version of CBASIC we reviewed has graphics capabilities. The graphics statements fall into five categories: output, format, viewing area, input, and control. The output statements and functions serve to plot lines and shapes with optional filling, mixing text with graphics.

The format statements select the color, character height, line style, text angle, marker height, and style. The viewing area statements define the aspect ratio of the *X* and *Y* axis of the graphics device, defining the graphics area boundary and the real-world scales. One statement in the input category, *GRAPHICS INPUT*, serves to input the *X* and *Y* coordinates using cursor control keys. This seems to be aimed at using a mouse, though the manual does not state it explicitly. The control type statements perform tasks such as clearing the screen, clipping and opening and closing graphics devices.

Storing data on files uses either sequential or random access records. Files must be first *CREATE*d before they can be *OPEN*ed. This differs from Microsoft BASIC where the *OPEN* statement will automatically create a new file if necessary. The *CREATE* statement may include options for the record length (in the case of random access records), the I/O mode, and the number of additional buffers to speed up I/O operations.

CBASIC uses a more direct approach in accessing random records. Numeric data is not packed into character codes. As a matter of fact, sequential and random access statements are identical except for the presence of a record number for the random access record type. Reading and writing single bytes is also implemented.

Error handling follows the familiar *ON-ERROR GOTO* scheme. CBASIC provides three error-identifying functions: two-letter error code, error line number, and 16-bit MP/M II extended error code.

CBASIC benchmark speed tests showed it keeping up with the BASCOM compiler series for the Sieve and sort tests. The matrix test took twice as much time, and the trig test was the slowest. This is due to the fact that the trig functions were developed using a formal language instead of assembly language. However, the use of trig functions in business is less common.

### Microsoft Business BASIC Compiler

Business BASIC Compiler was created by building on Microsoft Corp.'s own BASIC Compiler. Many powerful and versatile features were added, making the product a serious rival to the CBASIC compiler from Digital Research. For these reasons, we will limit our discussion to those added features.

Business BASIC Compiler allows separate compilation of modules and subprograms. This enables the creation of more structured programs and powerful general purpose libraries. More about this will come later.

This compiler also enables the user to use BASIC programs with either numbered program lines or labels. This is done by using a compiler switch typed on the DOS command line. Business BASIC Compiler allows both numeric and alphanumeric labels, while Microsoft BASIC Compiler limits itself to numeric labels only.

For the sake of maintaining higher math precision and less roundoff errors, Business BASIC Compiler applies BCD math and performs all floating point arithmetic and function evaluations in double pre-

Realia COBOL

What to do while your COBOL programs compile and execute:

1. Wait.
2. Wait some more.
3. Stop waiting. Call Realia.

Patience isn't always a virtue.
Realia COBOL is fast:

**Compilation Speed** (minutes:seconds)

| Lines in Program | Realia COBOL | mbp COBOL | Level II COBOL | R-M COBOL | Microsoft COBOL |
|---|---|---|---|---|---|
| 1,000 | :51 | 8:33 | 3:42 | 5:05 | 5:11 |
| 5,000 | 3:30 | 48:07 | 16:58 | * | 45:26 |

*Could not successfully compile the program.

**Execution Time Ratio**
(Gibson Mix; calculated S-Profile)

| Realia COBOL | mbp COBOL | Level II COBOL | R-M COBOL | Microsoft COBOL |
|---|---|---|---|---|
| 1.0 | 3.6 | 14.7 | 21.6 | 22.3 |

**Sieve of Eratosthenes**
0.818 seconds per iteration

All benchmark tests were performed on an IBM PC XT with 192KB of memory. IBM PC XT is a registered trademark of International Business Machines Corporation. mbp COBOL, of mbp Software and System Technology. Level II COBOL, of Micro Focus. R-M COBOL, of Ryan McFarland and Microsoft COBOL, of Microsoft.

Realia COBOL is written in COBOL. We offer you the tools we use ourselves:

- Our FOLLOW-THE-SOURCE™ interactive symbolic debugger. Works with normal native code.
- A speedy full-screen editor that handles very large files.
- Mainframe IBM VS COBOL compatibility.
- Interfaces to Assembler and C.
- No royalty or run-time fee.
- No limit on program size, up to available memory.
- In our new release, no need to insert the product diskette when you're using a hard disk.

Realia COBOL costs $995. Qualified companies can try it for free. Call us. And ask about our other products, Spacemaker™ and Termulator™.

What are you waiting for?

REALIA inc.
10 South Riverside Plaza
Chicago, Illinois 60606
(312) 346-0642
TELEX: 332979 (REALIA INC)

CIRCLE 76 ON READER SERVICE CARD

# 6 TIMES FASTER!

## SuperFast Software Development Tools

INCREASE YOUR PROGRAMMING EFFICIENCY
with high-performance software development products from SLR Systems.
**No other tools approach the speed or flexibility of the SLR Systems line.**

**"Z80ASM is an extraordinary product...",**
*Robert Blum, Sept. 84 DDJ*

**"...in two words, I'd say speed & flexibility",**
*Edward Joyce, Nov. 84 Microcomputing*

### ASSEMBLERS

- RMAC/M80 macros
- Nested INCLUDES & conditionals
- 16 char. labels on externals
- Built in cross-reference
- Optional case significance
- Phase/dephase
- Math on external words and bytes
- Define symbols from console
- Generate COM, HEX, SLR-REL, or Microsoft-REL files
- Time & Date in listing
- Over 30 configure options

| | |
|---|---|
| Z80ASM -full Zilog Z80 | $125 |
| *NEW!* Z80ASM+ -all tables virtual | $195 |
| *NEW!* SLRMAC -full Intel 8080, with Z80.LIB extensions internal | $125 |
| *NEW!* SLRMAC+ -all tables virtual | $195 |

Z80 CPU, CP/M compatible, 32K TPA required.

**"Z80ASM...a breath of fresh air...",**
*Computer Language, Feb. 85*

### LINKERS

- Links SLR & M80 format files
- Output HEX or COM file
- Three separate address spaces
- Load map and SID/ZSID .SYM file

SLRNK+ includes:

- All tables overflow to disk
- HEX files do not fill unused space
- Intermodule cross-reference
- EIGHT separate address spaces
- Works with FORTRAN & BASIC
- Generate PRL & SPR files
- Supports manual overlays
- Full 64K output

| | |
|---|---|
| SLRNK -fastest memory based | $125 |
| *NEW!* SLRNK+ -full featured virtual | $195 |

Combo Paks available from $199. - $299.

**For additional information contact SLR Systems**

1-800-833-3061, in PA (412) 282-0864
1622 N. Main St., Butler, PA 16001 • Telex 559215

C.O.D., Check or Money Order Accepted

**SLR Systems**

**CIRCLE 73 ON READER SERVICE CARD**

---

**OASYS**
**TOOLKIT**
**SNAP-SHOT**

# C-68000

## CROSS AND NATIVE OPTIMIZING COMPILERS

FOR 68000/10 (and 68020 SOON)

OASYS offers a *"ONE STOP SHOPPING"* service for software developers in need of proven 8-, 16- and 32-bit cross and native tools for Unix and non-Unix 68000, 8086 and 32000 systems. Our critically acclaimed and widely used 68000 tool kit offers high quality, reliable, cost-effective tools.

The OASYS 68000 tool kit consists of Green Hills compilers (C, Pascal and FORTRAN), our own M68000 Macro Assembly Development package, and dozens of other OASYS compatible support tools. Simply stated, we beat the competition on price, speed and tightness of emitted code.

### C-68000/10

- Full K & R with Western Electric and Berkeley extensions
- Complete run-time library available as source. No royalty if passed on.
- Supports DEC & IEEE Floating Point
- Integrated optimizer: 30% tighter code than Portable C; 4 times faster
- Generates M.I.T. or EXORmacs assembly source code
- Interfaces to all OASYS tools and Pascal, FORTRAN and PL/M-68K compilers
- Ideal for cross development of boards with no OS, a kernel OS (e.g. VRTX, PSOS, MTOS), or Unix based 68000's

### 68000/10 Assembly package

- EXORmacs compatibile Macro Assembler, Linker, Librarian, and Cross Reference Utility
- Generates S-records and a.o.
- PIC and reentrant code
- Used 2 years in house
- Over 3,000 sold to date
- Runs on VAX, Prime, PDP-11, 68000's, 8086/88 (PC)
- Written entirely in C

### Coming soon

- 68020 C and Cross assembler

### Other tools

- Symbolic C Source Debugger
- 68000 Simulator & Disassembler
- C Linecount and Time Profiler Utility (CLUE™)
- LINT for VAX/VMS
- Check Out compiler (SAFE-C™)
- Communications tools

**OASYS**
60 ABERDEEN AVENUE
CAMBRIDGE, MA 02138
(617) 491-4180

C68K-11/84

**CIRCLE 84 ON READER SERVICE CARD**

**Programmers:**
**This new book shows you how your software can support over 150 video terminals and microcomputers!**

A Programmer's Guide to Video Display Terminals
by David Stephens

How to support the wide variety of video display terminals has long been a problem for programmers. The "cursor up" code for one VDT might well clear the screen on another!

If you have spent time searching for control code sequences then this new book will be a welcome relief. We gathered 146 data sheets to give you a single source for your VDT support questions.

**A Programmer's Guide to Video Display Terminals** shows you how to clear the screen, position the cursor (with examples!), home the cursor, make seven erasures, turn video attributes on and off, and recognize cursor (arrow) keys for over 150 VDTs! We even include data for many VDTs which are no longer manufactured.

You will find this book an indispensible aid if you are a programmer, software developer, consultant, dealer, OEM, value-added retailer, or just frequently called on to support a variety of VDTs.

We are so sure that this book will eliminate your VDT support problems that we offer a FREE 15-day examination. To receive your copy, return the coupon below.

Examine it free for 15 days. If you are not completely satisfied, return it and owe nothing. Prepaid orders will receive a refund.

**15-DAY *FREE* EXAMINATION!**

*A Programmer's Guide to Video Display Terminals*
by David Stephens
Atlantis Publishing Corporation Dept. 203
P.O. Box 59467, Dallas, Texas 75229-1467
ISBN 0-936158-01-8 $30 335 pages, softcover

**Atlantis Publishing Corporation** Dept. 203
P.O. Box 59467, Dallas, Texas 75229-1467

☐ **YES!** Please send *A Programmer's Guide to Video Display Terminals* for 15 days FREE examination. If I decide to keep the book I will pay $30 plus shipping.

Texas residents add sales tax. Price subject to change. Offer subject to acceptance by Atlantis Publishing Corporation. Foreign buyers remit in US currency, specify method and add shipping.

Name ____________________

Company ____________________

Address ____________________

City, State, Zip ____________________

☐ Check or money order ☐ Bill me.
Publisher pays shipping on prepaid orders. Same return privilege.

☐ MasterCard ☐ Visa Exp. Date __________

Card Number ____________________

X ____________________

**SIGN HERE.** Credit orders invalid unless signed.

**CIRCLE 3 ON READER SERVICE CARD**

cision. Single-precision floating point constants and variables are converted into double-precision reals before entering in any mathematical operation.

Business BASIC Compiler allows static and dynamic arrays. Static arrays have their spaces allocated during compile time, while dynamic array spaces are allocated during run time. This allows arrays to be redimensioned by the program. This makes Business BASIC Compiler behave like the MS-BASIC interpreter and restores a feature not implemented in the Microsoft BASIC Compiler. Provisions are made to convert one type of array to the other.

This compiler implements multiline functions. No recursion is allowed, and no local variables exist. All arguments are passed by value. Business BASIC Compiler has implemented subprograms with arguments (passed by either value or reference) and both local and global variables.

Global variables work in two ways. First, the main program can declare global variables shared by all subprograms. Second, the subprograms can declare a list of variables shared with the main program. This minimizes some undesirable side effects brought about by the first method. Functions that return the lower and upper array bounds are also supplied to help create general purpose routines that handle arrays and matrices.

Business BASIC Compiler makes no secret about its competition with CBASIC. Many CBASIC keyboard, console, printer, and disk I/O statements as well as string manipulation functions are added for compatibility. Moreover, a special utility program is supplied to convert CBASIC source code into Business BASIC Compiler compatible code.

Business BASIC Compiler is pretty much as fast as CBASIC, with the exception of its faster evaluation of trig functions. It is as fast as BASCOM when performing the Sieve and sort tests.

### Microsoft BASIC Compiler, IBM BASIC Compiler, and MicroWay 87BASIC

The BASIC Compiler (BASCOM), a product of Microsoft Corp., is the compiler for MS-BASICA. Microsoft also worked with IBM to produce a BASCOM version for BASICA. 87BASIC, a product of MicroWay, is the version of IBM BASCOM that supports the 8087 numeric coprocessor chip. This allows compiled programs performing floating point calculations to work even faster that those compiled with BASCOM. The object code produced by these compilers is then linked to produce run-time .EXE program files.

BASCOM and 87BASIC process BASIC programs saved in ASCII form. The compilers allow parameters to be typed on the MS-DOS command line. These parameters instruct the compiler to perform a special function or alter a normal compiler function. This includes parameters for handling *ON ERROR GOTO* statements used for error trapping, event trapping, and interrupts associated with the *COM(n)*, *KEY(n)*, *PEN*, and *STRING(n)* statements and parameters enabling tracing compiled programs using the *TRON* and *TROFF* statements.

The same last parameter will check arithmetic overflow, underflow, array bounds, and coupled *GOSUB/RETURN* statements. In the event of run-time errors, it will display the corresponding BASIC line number. Another parameter produces code that does not depend on the run-time library BASRUN.EXE.

Finally, an interesting compiler parameter can view the BASIC program line numbers as labels. Thus they can be eliminated, except for those program lines referred to by *GOTO* and *GOSUB* statements.

The compilers allow metacommands or compiler directives. They must be included inside REMarks in the source code. They perform a variety of tasks such as including source code from other files, changing the maximum line width in the listing file, turning the listing of source or object code on and off, controlling pagination, and assigning a title and subtitle for the listing.

Compiling BASIC programs has its limitations. Not all BASIC programs can be compiled as they exist because some of the features offered by the BASIC interpreter are restricted or removed by the compiler.

Arrays must be dimensioned with integer constants. No variables are allowed, thus eliminating dynamic dimensioning and replacing it with rigid dimension limits. The source program must be edited and recompiled if larger arrays and matrices are needed.

Programming tricks that cause the program to *GOTO* inside *FOR-NEXT* and *WHILE-WEND* loops are not allowed. Chaining is more restricted. The *ALL* clause, which automatically passes all variables to the chained program, is not supported. Instead, the *COMMON* statement must be used to pass variables. Failing to mention any variable will cause its removal as chaining proceeds.

We used IBM BASCOM version 1.0 for the benchmark test. The test speeds are almost the same as the latest version of Microsoft BASIC compiler (version 5.38). The 87BASIC compiler showed identical timings for the Sieve and sort tests, which involve no floating points. However, when it came to the matrix and trig test, the advantages of the 8087 chip were evident.

### Softaid MTBASIC

Softaid Inc. offers the MTBASIC (multitasking BASIC) compiler for an

# RP/M T.M.

By the author of Hayden's "CP/M Revealed."

New resident console processor RCP and new resident disk operating system RDOS replace CCP and BDOS without TPA size change.

User 0 files common to all users; user number visible in system prompt; file first extent size and user assignment displayed by DIR; cross-drive command file search; paged TYPE display with selectable page size. SUBMIT runs on any drive with multiple command files conditionally invoked by CALL. Automatic disk flaw processing isolates unuseable sectors. For high capacity disk systems RDOS can provide instantaneous directory access and delete redundant nondismountable disk logins. RPMPIP utility copies files, optionally prompts for confirmation during copy-all, compares files, archives large files to multiple floppy disks. RPMGEN and GETRPM self-install RP/M on any computer currently running CP/M®2.2. Source program assembly listings of RCP and RDOS appear in the RP/M user's manual.

RP/M manual with RPMGEN.COM and GETRPM.COM plus our RPMPIP.COM and other RP/M utilities on 8" SSSD $75. Shipping $5 ($10 nonUS). MC, VISA.

118 SW First St. - Box G
Warrenton, OR. 97146

**Micro Methods, Inc.**

(503) 861-1765

CIRCLE 52 ON READER SERVICE CARD

# Now With Windowing!
## $49.95 Basic Compiler

## MTBASIC

Features:

| | |
|---|---|
| Multitasking | Windowing |
| Handles interrupts | Interactive |
| Fast native code | Compiles quickly |
| Floating point | No runtime fee |

MTBASIC is a true native code compiler. It runs *Byte's* Sept. '81 sieve in 26 seconds; interpreters take over 1400 seconds! Because MTBASIC is multitasking, it can run up to 10 Basic routines at the same time, while displaying ten separate windows. Pop-up/down menus are a snap to implement.

The MTBASIC package includes all the necessary software to run in interpreter or compiler mode, an installation program (so *any* system can use windowing), three demonstration programs and a comprehensive manual.

AVAILABLE for CP/M (Z-80), MS-DOS, and PC-DOS systems.

ORDERING: Specify format when ordering. We accept Visa, MC, checks and COD. Send $49.95 plus $3.50 shipping and handling ($10 overseas) to:

SOFTAID, Inc.

P.O. Box 2412 Columbia, MD 21045-1412
301/792-8096

CIRCLE 78 ON READER SERVICE CARD

OASYS
TOOLKIT
SNAP-SHOT

# WIZARD
# C-8086
## CROSS AND NATIVE OPTIMIZING COMPILERS
FOR 8086/87/88/186/286

As part of OASYS' *"ONE STOP SHOPPING"* service for software engineering tools, we are proud to announce the addition of WIZARD C to our integrated collection of more than 50 professional programming tools (e.g. compilers, assemblers, linkers, debuggers, simulators & translators) for M68000, Intel 8086/80186 and NS32000 micros.

**WIZARD C benchmarks (against Lattice, Microsoft) prove that it is, by far, the most advanced, full featured, fastest, tightest, optimizing C compiler now available for cross and native (PC) development. Here's why . . .**

### FEATURES

- Complete K&R implementation plus V7, III extensions
- Supports 8087 Floating Point
- Built-in **LINT**
- Long, medium, short models
- 190+ UNIX III functions — complete run-time library
- In-line assembly allowed
- 100+ extensive warnings/diagnostics
- Intel and Microsoft compatibility at source and object levels
- Written in C — easily ported
- Comprehensive bound documentation
- Supports DOS 2.0, 2.1, IBM/BIOS

### SUPPORT TOOLS

- Symbolic C Source Level Debugger (CDEBUG™)
- 100% Intel compatible structured Macro Cross Assembler, Linker/Locater and Librarian
- 8086 Simulator
- Floating point math package (40+ functions)
- C Time Profiler (CLUE™)
- Checkout compiler (SAFE-C™)
- Comm. utilities for up/down loading to MDS, TEK, Microtek

### AVAILABILITY

**NATIVE:** PC/XT/AT using MS/DOS, PC/DOS (Xenix soon)

**CROSS:** VAX/VMS, Bsd 4.1, 4.2, III, V; 8086's, 68000's (All Unisoft III, V; ports); Callan, Masscomp, Sun, Pyramid, dozens more . . .

Call for pricing, OEM, Site, Corporate, Source and Maintenance licensing information.

OASYS
60 ABERDEEN AVENUE
CAMBRIDGE, MA 02138
(617) 491-4180

W86CH-11/84

CIRCLE 75 ON READER SERVICE CARD

incredible $49.95. It is a peculiar animal, having characteristics of both a BASIC compiler and a BASIC interpreter, with a few of its own oddities thrown in.

To begin with, it has a built-in editor. Most unusual for a compiler, but it's very, very handy. MTBASIC supports screen windowing and recursion. Probably the most unusual feature of MTBASIC, however, is multitasking. The language has a time-slice scheduler which will manage up to 10 coresident BASIC programs. The scheduler will use either a periodic hardware interrupt or self-generated software interrupts for the slicing.

Another notable feature of this compiler is that it generates very fast code. The benchmarks prove this out. The code can be generated such that it is ROMable. In addition, it is possible to tie hardware interrupts into the program, allowing you to code interrupt service routines in BASIC.

The price for these nifty features seems to be compatibility. MTBASIC programs have a unique appearance. All variables must be declared (integer, real, or string) before they are used. Variables of different data types may not have the same name. For example, A and A$ cannot both be declared. No *END* statement is available, but you have *EXIT*, *STOP*, and *CANCEL* to choose from, depending on what you really want to do. The *INPUT* statement will not accept a prompt string. By now you should be getting the picture.

MTBASIC is a very different sort of BASIC. It programs differently, it has radically different features, and it is fast. It is also inexpensive to the point where I would just like to have it around for that occasional multitasking application. If you like your languages a little different, take a look at MTBASIC.

LATTICE® WORKS

LATTICE, INC. OFFERS C COMPILERS

The industry's standard C compiler, Lattice C, is now published directly by Lattice. You can obtain editions of Lattice C from other publishers, but when you purchase our edition, you get support directly from the people who wrote the C compiler. And when you register your purchase with Lattice, you are notified of all updates and enhancements. You will also be notified of the new C programming tools as they become available from Lattice.

When you are ready to purchase a C compiler, consider the source. Then call Lattice, Inc.

LATTICE OFFERS C COMPILER UPGRADES TO MICROSOFT USERS

Lattice announces the opportunity for users of the Microsoft MS-DOS C compiler to upgrade to the standard Lattice C and obtain a free copy of Lattice's new C-SPRITE symbolic program debugger.

According to Dave Schmitt, Lattice, Inc. President, "Until recently, the Microsoft C compiler was an early version of our compiler. Now that Microsoft has switched to their own compiler, we feel that purchasers of the earlier product deserve a low-cost way to remain compatible with the latest Lattice technology and the extensive third-party support for the Lattice product."

The upgrade kit costs $150 and includes Version 2.20 of the Lattice C compiler with its new two-color, typeset manual and the C-SPRITE program debugger. (The regular retail prices are $500 for the compiler and $175 for C-SPRITE.)

This offer to Microsoft C users expires July 31. Upgrade orders must be accompanied by the original Microsoft diskettes.

LATTICE C USERS' GROUP FORMED

An independent Lattice C Users' Group has recently been formed, and is headed by Bill Hunt, author of the book, *The C Toolbox*. The group will serve users of Lattice C, but membership is open to any C compiler users.

Members will receive a bi-monthly newsletter explaining C features and providing examples of their use, exploring new ways to use Lattice C, and will include a question and answer column. The newsletter will be supplemented by a disk containing source files, demo programs and library functions, and the best of the public domain C programs.

Lattice, Inc. will provide complete support for the new users' group by sharing new plans, discovered bugs, and "tech" tips.

Register your C compiler now to receive complete information on joining the Lattice C Users' Group.

**ASK ABOUT OUR "TRADE UP TO LATTICE C POLICY"**

*After purchase, return registration cards for free subscription to the "Lattice Works" newsletter and important information about the Lattice Users Group.*

Lattice, Inc.
P.O. Box 3072
Glen Ellyn, IL 60138
(312) 858-7950
TWX 910-291-2190

**International Sales Offices**
**Belgium:** Softshop. Phone: (32) 53-664875.
**England:** Round Hills. Phone: (0672) 54675.
**Japan:** Lifeboat Japan. Phone: (03) 293-2311.

CIRCLE 18 ON READER SERVICE CARD

### Sparry BASIC B

This interesting product is still in its infancy. Sparry Software Labs probably calls this a business compiler since it lacks floating point integer. Instead, all arithmetic is performed using 48-bit integers. As a result, the Sparry BASIC B compiler could neither compile nor run the trig benchmark. Array capability is also limited; only 256 elements per array are allowed, which forced major modifications to the Sieve and sort benchmarks. The Sparry compiler found 96 primes in the first 255 integers in 34 sec and sorted 255 integers in 25 sec. It was much better at the matrix benchmark turning in a terrific time of 1 min 9 sec.

If you look at the compile times, you will see they are very short. That is because the Sparry compiler does not compile down to an executable file. Rather it generates a small program file which is coupled with a run-time package.

One of the interesting features of this implementation is the use of virtual screens. Sparry supports up to four virtual screens, only one of which is displayed at any one time. Virtual screens may be swapped in less than a second for rapidly appearing screen updates. Another fine feature is direct support for ISAM file programming.

Sparry plans to upgrade this package in the future. Floating point numbers and operators will be added, and the array dimensions will be allowed to range to 32K. When these features are added (planned for version 1.3), this will indeed be a product worth considering.

### CP/M 86 INTERPRETER
### Digital Research Personal BASIC

This Digital Research Inc. BASIC implementation is a direct competitor of MS-BASIC in the CP/M 86 and concurrent CP/M 86 operating systems. The strong similarity and compatibility between the two BASICs is intentional.

If you know how to program in MS-BASIC then you know how to use Personal BASIC. This makes programs written in the former easily transportable to run with the latter. To save space we only focus on those aspects where Personal BASIC is different from MS-BASIC.

Personal BASIC is a true interpreter. It allows the user to enter syntax error-free programs by carrying an immediate check on each program line entered. The implementation offers a line editor to correct or change program lines. Personal BASIC can trace the execution of program lines

**MS-DOS, UNIX, Apple MAC, CP/M, NETWORKS and more. One c-tree ISAM DOES THEM ALL!**

c-tree™
BY FAIRCOM

2606 Johnson Drive
Columbia MO 65203

The company that introduced micros to B+ Trees in 1979 and created ACCESS MANAGER™ for Digital Research, now redefines the market for high performance, B+ Tree based file handlers. With c-tree™ you get:

- complete C source code written to K&R standards of portability
- high level, multi-key ISAM routines and low level B+ Tree functions
- routines that work with single-user and network systems
- no royalties on application programs

**$395 COMPLETE**

Specify format:
5¼" PC-DOS 3½" Mac
8" CP/M® 8" RT-11

for VISA, MC or COD orders, call
1-314-445-6833

Access Manager and CP/M are trademarks of Digital Research, Inc. 'MS is a trademark of Microsoft c-tree and the circular disc logo are trademarks of FairCom UNIX is a trademark of Bell Laboratories Apple is a trademark of Apple Computer, Inc.

©1984 FairCom

CIRCLE 34 ON READER SERVICE CARD

**TOTAL CONTROL:**

***FORTH:* FOR Z-80®, 8086, 68000, and IBM® PC**

**Complies with the New 83-Standard**

**GRAPHICS • GAMES • COMMUNICATIONS • ROBOTICS DATA ACQUISITION • PROCESS CONTROL**

- ***FORTH*** programs are instantly portable across the four most popular microprocessors.
- ***FORTH*** is interactive and conversational, but 20 times faster than BASIC.
- ***FORTH*** programs are highly structured, modular, easy to maintain.
- ***FORTH*** affords direct control over all interrupts, memory locations, and i/o ports.
- ***FORTH*** allows full access to DOS files and functions.
- ***FORTH*** application programs can be compiled into turnkey COM files and distributed with no license fee.
- ***FORTH*** Cross Compilers are available for ROM'ed or disk based applications on most microprocessors.

Trademarks: IBM, International Business Machines Corp.; CP/M, Digital Research Inc.; PC/Forth+ and PC/GEN, Laboratory Microsystems, Inc.

**FORTH Application Development Systems** include interpreter/compiler with virtual memory management and multi-tasking, assembler, full screen editor, decompiler, utilities and 200 page manual. Standard random access files used for screen storage, extensions provided for access to all operating system functions.

**Z-80 FORTH** for CP/M® 2.2 or MP/M II, $100.00; **8080 FORTH** for CP/M 2.2 or MP/M II, $100.00; **8086 FORTH** for CP/M-86 or MS-DOS, $100.00; **PC/FORTH** for PC-DOS, CP/M-86, or CCPM, $100.00; **68000 FORTH** for CP/M-68K, $250.00.

**FORTH + Systems** are 32 bit implementations that allow creation of programs as large as 1 megabyte. The entire memory address space of the 68000 or 8086/88 is supported directly.

| | |
|---|---|
| **PC FORTH +** | $250.00 |
| **8086 FORTH +** for CP/M-86 or MS-DOS | $250.00 |
| **68000 FORTH +** for CP/M-68K | $400.00 |

**Extension Packages** available include: software floating point, cross compilers, INTEL 8087 support, AMD 9511 support, advanced color graphics, custom character sets, symbolic debugger, telecommunications, cross reference utility, B-tree file manager. *Write for brochure.*

VISA

***Laboratory Microsystems Incorporated***
*Post Office Box 10430, Marina del Rey, CA 90295*
*Phone credit card orders to (213) 306-7412*

MasterCard

CIRCLE 35 ON READER SERVICE CARD

**ACTIVE TRACE**

**"Software that lives up to its promises.** *When a Basic program doesn't work the way you want it to, this package... will help you track the problem down... Scope is a tool for the beginning, advanced, or professional programmer, and it begins where the cross reference maps leave off."*
**Howard Glosser, *Softalk for the IBM Personal Computer* July '84, pp 120-121**

***"Extremely useful program...*** *Anyone doing much programming in Basic should appreciate Active Trace a lot."*
**Jerry Pournelle, *Byte Magazine* April '83, p 234**

***"A marvelous Basic programming aid...*** *It's just amazing to watch a program you wrote run under Scope, and debugging becomes if not trivial, then at least doable"*
**Thomas Bonoma, *Microcomputing,* Dec. '83, p 22**

*"...**a really neat utility**... designed to untangle even the most convoluted Basic program.... The documentation is almost worth the price of the package."*
**Susan Glinert-Cole, *Creative Computing,* July '84, p 210**

**Active Trace** will lead you through your program letting you know variable values (all variables or just those you specify) as they change. Your program's internal activity is presented on your screen, or printer, or it can be saved on disk. **It's simple, effective and works with the BASIC you already own.**

**Active Trace $79.95**
Includes Scope, XREF mapping and documentation

**Active Trace** is available for most **MS-DOS** and **CPM 2.2** systems and supports the special features of Brand specific versions of Microsoft Basic such as **Basica** on the **IBM-PC**.

AWARECO
Active Software

P.O. Box 695 Gualala, CA 95445
(707) 884-4019
800-358-9120(US) 800-862-4948(CA)

Active trace, Active software, and Scope are trademarks of AWARECO—CPM is a trademark of Digital Research—MS-DOS and Microsoft are trademarks of Microsoft Corporation—IBM-PC is a trademark of IBM Corp.

CIRCLE 1 ON READER SERVICE CARD

**DeSmet C**

**8086/8088 Development Package $109**

**FULL DEVELOPMENT PACKAGE**
- Full **K&R** C Compiler
- Assembler, Linker & Librarian
- Full-Screen Editor
- Execution Profiler
- Complete **STDIO** Library (>120 Func)

**Automatic DOS 1.X/2.X SUPPORT**

**BOTH 8087 AND S/W FLOATING POINT**

**OVERLAYS**

**OUTSTANDING PERFORMANCE**
- First and Second in AUG '83 BYTE benchmarks

**SYMBOLIC DEBUGGER $50**
- Examine & change variables by name using C expressions
- Flip between debug and display screen
- Display C source during execution
- Set multiple breakpoints by function or line number

**DOS LINK SUPPORT $35**
- Uses DOS .OBJ Format
- LINKs with DOS ASM
- Uses Lattice® naming conventions

Check: ☐ Dev. Pkg (109)
☐ Debugger (50)
☐ DOS Link Supt. (35)

SHIP TO: ____________

____________

____________ ZIP ______

**C WARE CORPORATION**

**P.O. BOX C**
**Sunnyvale, CA 94087**
**(408) 720-9696**

All orders shipped UPS surface on IBM format disks. Shipping included in price. California residents add sales tax. Canada shipping add $5, elsewhere add $15. Checks must be on US Bank and in US Dollars. Call 9 a.m. – 1 p.m. to CHARGE by VISA/MC/AMEX.

Street Address: 505 W. Olive, #767, (94086)

**CIRCLE 11 ON READER SERVICE CARD**

by either displaying the entire line or just its number, in addition to keeping track of the contents of variables.

The Personal BASIC interpreter ran faster than its look-alike rival, MS-BASIC, except in the matrix test. The test was performed using Concurrent PC-DOS, a multitasking operating environment produced by Digital Research.

## CP/M INTERPRETERS

We took Microsoft Corp.'s extended disk BASIC interpreter, MS-BASIC, as the prototypical yardstick for comparison purposes here. Among the five BASICs we review, two are compilers, and three are interpreters.

Two of the interpreted languages, DeltaBASIC and DBASIC, are geared specifically to data base programming. Of the two, DBASIC is clearly the most powerful and efficient. The last interpreter, BaZic, is a rudimentary language aimed at novice users.

The two compilers, Multi-BASIC and BASIC/Z, were both quite good, but Multi-BASIC had better speed on the benchmarks and offers more programming versatility. For serious general programming, it would be the winner among the CP/M languages we looked at. As was to be expected, the compilers generally did much better than the interpreters on the benchmarks, although DBASIC did very well. The Sieve program gave every language problems, except Multi-BASIC. The BASIC/Z and Multi-BASIC compilers did surprisingly poorly on the trig tests. The tests were run on a Kaypro II with 2.5 MHz clock speed.

### DeltaSoft DeltaBASIC.

DeltaSoft Inc.'s DeltaBASIC is an interpreted BASIC modeled after IBM/34 BASIC and is self-proclaimed as a business-oriented BASIC. It is actually a fairly weak subset of MS-BASIC, with the addition of two new file types: keyed sequential and keyed random. Additionally DeltaBASIC offers extended formatting of data in I/O and a utility to design data entry screens that may be used by DeltaBASIC programs. The product is principally intended for the commercial BASIC programmer interested in constructing data bases.

The interpreter functions somewhat like the MS-BASIC interpreter. Source programs may be written with the line editor and saved in either tokenized or ASCII format. No built-in *RENUM* feature is available, but a utility program is supplied to do this job. There is *AUTO* line generation. Also, no *TRACE* for debugging exists. The control structures are the generic BASIC ones; there is no *REPEAT-UNTIL* or *WHILE-WEND*.

Error handling is the *ON-ERROR-GOTO* variety, with a limited set of error conditions. The only data types are real (single and double) and string (255 characters). Rather than *MID$*, *LEFT$*, *RIGHT$*, and *INSTR*, there is only one string function, which returns substrings at specified locations. Chaining can occur with common variables and arrays.

The distinctive features of DeltaBASIC are its provisions for reading and writing data files and its two new types of data file besides the usual sequential and random, keyed sequential and keyed random. A *FORM* statement can be used to specify the record fields for all I/O operations, and extensive formatting is possible, including a COBOL-like PIC.

Using the data entry screen utility, you can design screens (for supporting terminals) with fields having low-intensity, reverse video, blinking, protection, underline, and using the function and arrow keys. These screens can then be accessed by your program for data I/O.

Keyed sequential and keyed random files are really two files: a data file and an associated sorted key file (the key file contains the key tokens for one selected record field in the data file and pointers to the records in the data file which match a particular key token). When you read a data file in keyed sequential mode, the sorted key file is read to determine the order of the sequential reads (really ordered random reads) in the data file. When you read or rewrite in keyed random mode, the sorted key file is read first to determine the random record number of the data item associated with a particular key token. DeltaBASIC allows only one key per data file. In TurboDos and MP/M, record locking is possible.

In addition to the utilities previously mentioned, utilities are available for sorting keyed data files (sorting the key file, actually, in ascending order), converting ASCII text files to data files (and back again), removing records from data files, converting unkeyed data files to keyed data files, and generating help messages for data entry screens.

The manual is rather sparse and inadequate, though it does go to some effort to explain the use of keyed data files and the use of the utilities. Run-time facilities are available for distribution programs. DeltaBASIC's chief selling point is the easy creation of moderately sophisticated data bases. DeltaBASIC did the poorest of all the CP/M BASICs on the benchmarks. It couldn't dimension an array large enough to run the Sieve test and has no COS function. On the other tests its times were slow.

### Micro Mike BaZic

Micro Mike's BaZic was first designed as a NorthStar compatible BASIC interpreter under NorthStar DOS and then extended to the CP/M operating system. With that origin, the interpreter bears more resemblance to NorthStar BASIC

FINALLY...

THE C JOURNAL

**The C Journal** will help YOU use C on YOUR machine —IBM PC™, CP/M™, Macintosh™, or UNIX™-based — micro, mini, or mainframe.

It's the ONE publication for programmers, software managers, and other computer professionals who need to keep aware of developments in the industry's fastest-growing language.

- regular columns for novice through advanced C programmers
- software product and book reviews
- tips on working with major compilers and operating systems
- news from the ANSI standards committee and the industry

For **FREE** sample issue and discount subscription information, write, call, or circle our reader service number. **The C Journal** is a quarterly publication, and costs $28/year (add $9 for overseas airmail).

Subscriptions/Advertising:
Christina Gardner
(201) 989-0570

Editorial:
Rex Jaesche
(703) 860-0091

another independent publication from
**InfoPro Systems**
3108 Route 10
P.O. Box 849
Denville, NJ 07834

MasterCard
VISA

CIRCLE 5 ON READER SERVICE CARD

C Sick?

PLZ is the cure!

**Introducing a native code PLZ compiler for the 68000, featuring:**

☐ Complete PLZ language, including structure assignment and comparison
☐ Fully compatible with Zilog Z80, Z8000 PLZ
☐ Ideal for embedded, ROM based systems
☐ Strongly typed
☐ Data types include signed and unsigned byte, word and longword
☐ All of the protection of Pascal, with the flexibility of C
☐ Inherently more portable than either Pascal or C
☐ Easy for Pascal or C programmers to learn
☐ Fully compatible with the CP/M-68K C library

Requires CP/M-68K. Other systems and CPU's supported soon.

**Package includes:**
68000 Compiler-Code generator
User Manual
Springer-Verlag "Report on the Programming Language PLZ/SYS"
One Year free updates

**All this for the low introductory price of** $75

Add $3 S/H. NJ Residents include 6% sales tax.

**KCSystems**
20 Lamington Drive, Succasunna, NJ 07876
(201) 927-9104

CIRCLE 45 ON READER SERVICE CARD

PRESENTING THE MEGAMAX C COMPILER

NEW FOR THE MACINTOSH

FEATURING:
• IN-LINE ASSEMBLY • ONE PASS COMPILATION • SUPPORT OF DYNAMIC OVERLAYS • FULL ACCESS OF MACINTOSH TOOLBOX ROUTINES • AND MUCH MORE ...

DEVELOPMENT SYSTEM PACKAGE INCLUDES:
• FULL-SCALE IMPLEMENTATION (K&R) C COMPILER • THE STANDARD C LIBRARY • ROM ROUTINES LIBRARY • LINKER • LIBRARIAN AND DOCUMENTATION ...

DEALER AND USER GROUP INQUIRES INVITED

$299.95

FOR MORE INFORMATION OR TO ORDER CALL OR WRITE:

Megamax, Inc.
BOX 851521, DEPT. 1
RICHARDSON, TX 75085-1521
(214) 987-4931

MACINTOSH IS A REGISTERED TRADEMARK OF APPLE COMPUTER INC.

CIRCLE 13 ON READER SERVICE CARD

ConIX™

*NOW ONLY $79.95!*

If you think you're missing out on innovative software developments because nobody is writing for CP/M™-80, take a look at us. We've adapted UNIX™ features to CP/M like never before, and with the kind of professional, quality-controlled product that you deserve. That product is none other than the critically acclaimed ConIX Operating System.

ConIX can provide any 48K+ CP/M-80 or compatible system with I/O Redirection and Pipes (uses memory or disk), perfected User Areas, Command and Overlay Path Searching, Auto Screen Paging, 8Mb Print Buffering, 22 new SysCalls, Function Keys, "Virtual" disk system, Archiver (saves over 50% disk), extensive command language, 300+ variables, 100+ commands, pull-down menu, and much more! Uses as little as 1/2K RAM! Runs with CP/M for true data and software compatibility. Installs easily without any system mods!

The ConIX package lists at $165 and has been advertised and sold internationally to many enthusiastic customers since October 1983. As a special limited offer, we've lowered the price of the complete ConIX system by 50% to only $79.95! Don't miss this opportunity to bring your 8-bit micro back into the software revolution. Order your copy of ConIX today!

Price includes manual, 8" disk, and user support. 5¼" conversions available. Contact your local dealer, or buy direct and add shipping: $4.50 UPS, $10 Canada, $25 overseas. NY residents add sales tax.

**Computer Helper Industries Inc.**
P.O. Box 680 Parkchester Station, NY 10462
Tel. (212) 652-1786 (for information/orders)

*"We're helping your computer work better for you!"*

UNIX: AT&T Bell Labs, CP/M: Digital Research, ConIX: Computer Helper Ind.

CIRCLE 10 ON READER SERVICE CARD

than to MS-BASIC and, in comparison to MS-BASIC, is a decidedly weaker language.

Source programs may be prepared with the line editor, are saved tokenized, require line numbers, and can't utilize labels. The CP/M package comes with a conversion utility to translate ASCII files to tokenized BaZic files and back again, so that it is possible to use your editor or word processor to prepare or revise programs. There is *RENUM*bering and *AUTO*matic line generation but no debugging facility like *TRACE*.

BaZic has only two data types—strings and reals—and it permits only real arrays (the BaZic manual recommends kludging this for strings by dimensioning a string large enough to serve as an array). The interpreter is actually supplied in four versions, according to the amount of precision you want in your reals: 8, 10, 12, or 14 places.

Arithmetic is floating point, although file numerics are saved BCD. String lengths are unlimited but require dimensioning. There are no intrinsic string functions except conversion to numeric and length. Control structures are vanilla BASIC (*IF-THEN-ELSE*, *GOTO*, *GOSUB*, *FOR-NEXT*—but you can prematurely *EXIT* a loop). Like CP/M MS-BASIC, there are no procedures; the one syntactic advantage BaZic has over MS-BASIC is that it permits multiline, user-definable functions.

BaZic supports both sequential and random files but not record structures. Error trapping is of the localized *ON ERROR-GOTO* sort. There is a function to make CP/M DOS calls. In general, BaZic resembles a slightly primitive MS-BASIC, with many of the same functions but a moderately different syntax.

BaZic seems to have the intent of being a novice's language. Half of the manual is devoted to the beginning programmer. In that purpose it is fairly successful, but it is hardly a language for serious program development. (Incidentally, some of the manual is incomplete, according to the table of contents.) BaZic performed only moderately well on the benchmarks, except for the matrix test, where it turned in a time comparable to the BASIC/Z compiler.

### Spectrum Logic DBASIC.

Like DeltaBASIC, Spectrum Logic Corp.'s DBASIC is really a support language for data base design and maintenance. Modeled directly after MS-BASIC, it is syntactically and operationally a clone of that language, except for its file-handling capabilities. These capabilites are much expanded, beyond the standard sequential and random, to include keyed indexed sequential files and DBASIC's version of CP/M random files. They are here termed direct access files and can be organized into volumes of associated files under the management of one key file.

Since DBASIC is virtually identical to CP/M MS-BASIC, except for the new file types, there is little point in reviewing the features of that language here. Anyone familiar with MS-BASIC will be instantly at home with DBASIC (one noteworthy difference exists, though—DBASIC has no chaining).

The appeal of DBASIC is in its potential as a data base maintenance tool, which we can only hint at here. Going a step beyond DeltaBASIC's treatment of key files, DBASIC's indexed keyed, sequential files organize the key records hierarchically, according to which data field a particular key token corresponds to in the data file (direct access file) record. Thus, keys are not limited to a single field in the data file record, but the user can key on any or all fields. This amounts to allowing multiple keys, at the cost of some redundancy in the key file, and it permits a wide range of sorting and accessing techniques.

As tools for using the indexed keyed files and their associated direct access files, there are a variety of new commands. Commands for the keyed index files are: *INITI*, *OPENI*, *CLOSEI*, *SEEK*, *ADVNCE*, *INSERT*, *FORCE*, *UPDATE*, *DELETE*, *STATI*. Commands for the direct access files are: *INITD*, *OPEND*, *CLOSED*, *ALLOC*, *RELESE*, *READ*, *WRITE*, *STATD*. In addition to these, optional utilities are available for specialized arithmetic and string handling. All of these features, embedded in the MS-

*CP/M-80 C Programmers . . .*

**Save time**

. . . with the BDS C Compiler. Compile, link and execute *faster* than you ever thought possible!

If you're a C language programmer whose patience is wearing thin, who wants to spend your valuable time *programming* instead of twiddling your thumbs waiting for slow compilers, who just wants to work *fast*, then it's time you programmed with the BDS C Compiler.

BDS C is designed for CP/M-80 and provides users with quick, clean software development with emphasis on systems programming.

BDS C features include:

- Ultra-fast compilation, linkage and execution that produce directly executable 8080/Z80 CP/M command files.
- A comprehensive debugger that traces program execution and interactively displays both local and external variables by name and proper type.
- Dynamic overlays that allow for run-time segmentation of programs too large to fit into memory.
- A 120-function library written in both C and assembly language with full source code.

Plus . . .

- A thorough, easy-to-read, 181-page user's manual complete with tutorials, hints, error messages and an easy-to-use index — it's the perfect manual for the beginner *and* the seasoned professional.
- An attractive selection of sample programs, including MODEM-compatible telecommunications, CP/M system utilities, games and more.
- A nationwide BDS C User's Group ($10 membership fee — application included with package) that offers a newsletter, BDS C updates and access to public domain C utilities.

Reviewers everywhere have praised BDS C for its elegant operation and optimal use of CP/M resources. Above all, BDS C has been hailed for it's remarkable *speed*.

*BYTE* Magazine placed BDS C ahead of all other 8080/Z80 C compilers tested for fastest object-code execution with all available speed-up options in use. In addition, BDS C's speed of compilation was almost *twice* as fast as its closet competitor (benchmark for this test was the Sieve of Eratosthenes).

"I recommend both the language and the implementation by BDS very highly."
Tim Pugh, Jr.
in *Infoworld*

"Performance: *Excellent.*
Documentation: *Excellent.*
East of Use: *Excellent.*"
*InfoWorld*
Software Report Card

". . . a superior buy . . ."
Van Court Hare
in *Lifelines/The Software Magazine*

***Don't waste another minute on a slow language processor. Order your BDS C Compiler today!***

Complete Package (two 8" SSSD disks, 181-page manual): **$150**
Free shipping on prepaid orders inside USA.
VISA/MC, COD's, rush orders accepted.
Call for information on other disk formats.

BDS C is designed for use with CP/M-80 operating systems, version 2.2 or higher. It is not currently available for CP/M-86 or MS-DOS.

BDSoftware

BD Software, Inc.
P.O. Box 2368
Cambridge, MA 02238
(617) 576-3828

CIRCLE 4 ON READER SERVICE CARD

BASIC-like environment, add up to make an efficient data base management tool with much more versatility than DeltaBASIC.

The manual is extensive but rather poorly done—very obviously written by mainframe data base programmers who live in their own world. However, there was one humorous note; in discussing assembly language calls they say, "No specific method of loading the assembly language program into the memory space is suggested by this document. The method used depends to a large extent on the operating system and the skill of the user. This statement is a cop-out."

DBASIC did rather well in the benchmark tests, although it couldn't match the speed of the compilers. It was the only language that managed a fair showing on the trig tests. However, it ran the Sieve test slowly, as did all the BASICs except Multi-BASIC.

## CP/M COMPILERS

### Alcor Multi-BASIC.

Alcor System Inc.'s Multi-BASIC is the most comprehensive BASIC for CP/M that we reviewed. It is a p-code compiled BASIC with extensive development facilities. Alcor freely allows you to link the run-time facility with your object code to produce .COM files for commercial distribution, requiring only some documentation of Alcor's run-time code.

The compiler generates p-code object files in hex, which then can be directly executed with the run-time program RUNB or can be linked with the run-time support to produce stand-alone .COM files. The package also contains an optimizer to reduce p-code size and a program called CODEGEN, which produces Z80 machine code. CODEGEN can also output extended 8080 assembly source code, which can then be hand optimized and assembled with Alcor's XASM assembler (which is unfortunately not supplied with the BASIC package). All Multi-BASIC files can be linked by the LINKLOAD program. This allows you to blend p-code and machine code for maximum size and efficiency. Furthermore, Multi-BASIC p-code is compatible with Alcor's other languages, among them C and Pascal, so that programs written in one can call programs or routines in another.

Multi-BASIC supports compiled libraries, external procedures and functions, the *%INCLUDE* compiler instruction and the ability to combine p-code with machine code in linked object files. Among the external statements provided are: *CALLASM*, to call an assembly language routine that can be linked to the main program with LINKLOAD; *CALLBIOS*, to make CP/M BIOS function calls; and *CALLDOS*, to make CP/M operating system calls. *GETC* and *PUTC* are functions for fast byte sequential I/O.

Since the linking loader supports library search, the language is highly extensible by external procedures and functions. A second, overlayed version of the compiler is supplied to permit compilation of larger programs, and a hex-to-binary conversion program is supplied, which reduces the size of library files.

The Multi-BASIC compiler has switches that permit it to directly compile programs written in either Microsoft's MS-BASIC or Digital Research's CBASIC so that program conversion for these languages is unnecessary. Multi-BASIC's own syntax does not differ radically from MS-BASIC except for having extended statements and functions.

Program source code is prepared in ASCII using a text editor, and the Multi-BASIC package includes a full-screen editor, BLAISE, that has many of the features of a full word processor. (However, you might prefer to use your own word processor for program preparation, since the BLAISE commands take some getting used to, and it requires extensive configuration to utilize all of its capabilities. I would rather have seen the XASM assembler supplied than BLAISE). Multi-BASIC programs do not require line numbers and permit mnemonic labels for branching instructions.

The compiler is fast and efficient and furnishes statistics on stack and heap management. These can be reallocated at compile, link, or run times at the programmer's discretion. The compiler uses

**Advanced Screen Management made easy**

Now a professional software tool from Creative Solutions.

**WINDOWS FOR C™**

More than a window display system, WINDOWS FOR C is a video tool kit for all screen management tasks.

- Pop-up menus and help files
- Auto memory management
- Keyboard interpreter
- Word wrap
- Auto scroll
- Highlighting
- Color control
- Overlay and restore
- Plus a library of over 50 building block subroutines

**Designed for enhanced portability.**
**Easy to learn, easy to use.**

Once you've tried WINDOWS FOR C, you'll wonder how you ever managed without it.

Full support for IBM PC/XT/AT and compatibles, plus interfaces for non-IBM computers; Lattice C, CI-C86, Mark Wm. C, Aztec C, Microsoft C, DeSmet C (PC/MSDOS),

**NEW** Ver. 3.1
**Enhanced portability.**
**Topview compatible.**

WINDOWS FOR C $195
(specify compiler & version)
Demo disk and manual $ 30
(applies toward purchase)
**Full source available.**
**No royalties.**

**Creative Solutions**
21 Elm Ave., Box T4,
Richford, VT 05476
**802-848-7738**
Master Card & Visa Accepted
Shipping $2.50
VT residents add 4% tax.

CIRCLE 14 ON READER SERVICE CARD

static string allocation (no garbage collection). The run-time facility offers selective tracing for debugging if line numbers are used.

As mentioned, Multi-BASIC is a superset of CP/M MS-BASIC. Thus many of its features will be familiar to the MS-BASIC user. The data types are integer, single- and double-precision reals, and strings, with arrays of each type. String size is limited by available memory; variable name length is similarly unlimited. Multi-BASIC also recognizes binary, octal, and hex constants as integer types. Program control offers extended *IF-THEN-ELSE/ENDIF* and *WHILE-WEND*.

Error handling includes reporting operating system errors, *RESUM*ing, and reporting the line number of the error. File handling and I/O are essentially the same as in MS-BASIC. Where Multi-BASIC improves substantially on MS-BASIC is in allowing procedures and multiline user-definable functions, so that it is much easier to maintain program modularity. The manual, which comes in a binder with case, is thorough and well-organized.

Multi-BASIC did exceptionally well on the benchmarks— especially the sort routine—except for the trig tests, where it fell down rather badly, taking about 12 min for each routine. The tests were run using the optimizer and native code generator since those features were available. The compiled .COM files were rather large, as Multi-BASIC adds about 20K or so of run-time facilities.

New Release

C + C UTILITY LIBRARY = PRODUCT

• We have over 300 complete, tested, and, documented functions. All source code and demo programs are included.

• The library was specifically designed for software development on the IBM PC, XT, AT and compatibles. There are no royalties.

• Over 95% of the source code is written in C. Experienced programmers can easily "customize" functions. Novices can learn from the thorough comments.

*We already have the functions you are about to write*

**Concentrate on software development—not writing functions.**

THE C UTILITY LIBRARY includes:
• Best Screen Handling Available • Windows • Full Set of Color Graphics Functions • Better String Handling Than Basic • DOS Directory and File Management • Execute Programs, DOS Commands and Batch Files • Complete Keyboard Control • Extensive Time Date Processing • Polled ASYNC Communications • General DOS BIOS gate • Data Entry • And More •

• The Library is compatible with: Lattice, Microsoft, Computer Innovations, Mark Williams and DeSmet. Available Soon: Digital Research, Aztec and Wizard.

*C Compilers:* Lattice C—$349, Computer Innovations C86—$329; Mark Williams C—$449.

*C UTILITY LIBRARY* $185. Special prices on library & compiler packages.

Order direct or through your dealer. *Specify compiler* when ordering. Add $4.00 shipping for UPS ground, $7.00 for UPS 2-day service. NJ residents add 6% sales tax. Master Card, Visa, check or P.O.

ESSENTIAL SOFTWARE, INC
P.O. Box 1003 Maplewood, New Jersey 07040 914 762-6605

CIRCLE 33 ON READER SERVICE CARD

**System/z BASIC/Z.**

This native code compiler BASIC from System/z Inc. is not quite as versatile as Alcor's Multi-BASIC but has some unique recommendations of its own. It can be run as an interactive compiler, which checks each line entered for correct syntax. Also, source files can be prepared with your editor or word processor. It will read the source programs of other versions of BASIC and mark syntax errors as REM lines for later editing. Source programs require line numbers, but you can use labels for branching instructions. The built-in line editor has global search and replace for variables. Debugging is by tracing selected lines, with optional display of up to four variables and optional single-stepping.

The compiler uses a run-time package that may be linked to a main program and shared by chained programs. (System/z allows free distribution of the run-time module, linked, but requires a copyright notice flashed on the screen on start-up of your programs.)

The compiler doesn't support libraries with externals but does have the *%INCLUDE* option for including other source files (no file extraction). The compiler and run-time module can be installed for a variety of terminals and printers and, where appropriate, will support cursor addressing, reverse video, underline, blinking, high intensity, and double-width and double-height printer characters.

Distribution programs can be similarly installed with the *INSTALL* utility supplied. The installation process is lengthy but undemanding. A nice touch is that the interactive compiler has commands for converting among binary, octal, decimal, and hex constants.

The language has five data types: bytes, words, integers, reals, and strings. Arrays for each type can be dynamically allocated. One of the unusual touches of the language is built-in sorting and searching for any type of array. All integer and real arithmetic is BCD. Storage for integers and reals can be declared with a *SIZES* statement, with precision for reals to 18 places.

The storage for strings is statically allocated with the maximum being 250 characters, although you can in effect manipulate longer strings as byte arrays. The language recognizes binary, hex, and octal constants, and Boolean values.

*IF-THEN-ELSE*s are limited to one line, but the language has *DO-UNTIL*, *WHILE-WEND*, and *WHEN-WHEND* to supplement the usual *GOTO*s and *GOSUB*s. The language lacks procedures but does have recursive, multiline user functions.

A nice feature is that you can automatically load assembly-language sub-

## OVERCOME
## FORTRAN and PASCAL
## LIMITATIONS WITH

$89 Visa/MC

## NO LIMIT

A library of 58 Assembler routines transforms MS FORTRAN and PASCAL plus other 8086/87/88 FORTRANs into the flexible, responsive, complete language needed for the microcomputer environment. Ver 1.0 Features:

| | |
|---|---|
| **EXTENSIVE GRAPHICS** | (Get, Put, Paint, Color, Dot, Line, Box, Circle, Ellipse, Large Characters) |
| **FULL SCREEN CONTROL** | (Windows, Cursor, Read/Write Screen) |
| **STRING MANIPULATION** | (Match, Compare, Concatenate/Extract, Pack, Justify, Zero Fill) |
| **KEYBOARD CONTROL** | (Read Key During Execution, String Read) |
| **FILE MANAGEMENT** | (Exist?, Rename, Delete) |
| **COMMUNICATIONS** | (Set Com Line, Send/Receive, Line/Modem Status) |
| **OTHER FEATURES** | (Peek, Poke, Determine Time/Date, Random Numbers, Beep, Clear Screen, OR/AND/XOR/NOT/NEG of Byte/Word, Printer Status) |

Ver 2.0 with 92 routines now available.
**Directories, Command Line Read, Program Chaining, Interrupt Driven Communications.** $129. Upgrade $40.

M | E | F Environmental Inc.
P.O. Box 26537 Austin, Texas 78755
**(512) 251-5543** **Outside Texas (800) 562-9700**

**CIRCLE 55 ON READER SERVICE CARD**

---

## the source debugger for lattice C

Your time and convenience come first! The MSD C Debugger™ is the last, and perhaps final, word in programming assistance for Lattice C users. C Debugger produces a high level view of C programs via function names, line numbers, variable names and C data types, plus a low-level view of machine addresses and instructions for testing assembler language functions.

**More features include:**
- All documentation is prepared for programmers.
- Online help screen throughout the process.
- Capability to single step through your program.
- Set break points, examine registers and variables.

**$165.00 + $3.50 shipping**

VISA MasterCard

To order, call or write:
MSD
**MICRO-SOFTWARE DEVELOPERS, INC.**
214½ W. Main St. • St. Charles, IL 60174
312/377-5151

Lattice C is a trademark of Lattice, Inc.

**CIRCLE 54 ON READER SERVICE CARD**

---

## WRITE

The Writer's Really Incredible Text Editor lives up to its name! It's designed for creative and report writing and carefully protects your text. Includes many features missing from WordStar, such as sorted directory listings, fast scrolling, and trial printing to the screen. All editing commands are single-letter and easily changed. Detailed manual included. Dealer inquiries invited. WRITE is $239.00.

## BDS's C Compiler

This is the compiler you need for learning the C language and for writing utilities and programs of all sizes and complexities. We offer version 1.5a, which comes with a symbolic debugger and example programs. Our price is (postpaid) $130.00.

## Tandon Spare Parts Kits

One door latch included, only $32.50.
With two door latches $37.50.
Door latches sold separately for $7.00.

All US orders are postpaid. We ship from stock on many formats, including: 8", Apple, Osborne, KayPro, Otrona, Epson, Morrow, Lobo, Zenith, Xerox. Please request our new catalog. We welcome COD orders.

Workman & Associates
112 Marion Avenue
Pasadena, CA 91106
(818) 796-4401

WORKMAN & ASSOCIATES
USER-FRIENDLY SOFTWARE

**CIRCLE 68 ON READER SERVICE CARD**

---

## The Tools You Need To C You Thru.

**Now the *WizardWare*™ Applications Programmers Toolkit provides everything you need to increase your C programming productivity.**

**APT™ features include:**
- COMPLETE SOURCE CODE (over 5000 lines!)
- File handling with direct & keyed access
- Screen and Report Generators, with full screen handling for your programs
- Generic Terminal Driver for portable code
- String math functions, and string manipulation routines
- Reference Manual on Disk (over 50 pages)
- Tutorial Manual (over 25 pages) with Source for Mailing List Manager
- A host of useful Utilities, Database and File Editors
- Available for Lattice C, Mark Williams C, DeSmet C, BDS C, others.

**Also Available: C-STARTER Toolkit, great for learning C!! Includes: Customized APT, DeSmet C Compiler, and "Programming in C on the IBM-PC" (200 pages)**

| | |
|---|---|
| APT/MS-DOS versions | $495 |
| APT/DeSmet C version | $395 |
| APT/BDS C version | $395 |
| C-Starter (binary APT, DeSmet Compiler and Book) | $295 |
| APT/Manual only* | $ 50 |

**Detailed Brochures on request**

*Manual Cost will be applied if APT purchased within 30 days ($10 re-stocking charge.) U.S. funds only, please.

Trademarks: MS-DOS/Microsoft, Lattice C/Lattice, Inc., MWC86/Mark Williams Co., DeSmet C/C Ware, CI-C86/Computer Innovations, Inc., BDS C/BD Software, DR-C/Digital Research, WizardWare, APT, C-Starter/Shaw ☆ American Technologies.

**Call (502) 583-5527**
Ask for APT™ or C-Starter, or Send Check to:
**Shaw ☆ American Technologies**
*WizardWare*™
**830 South Second St. - Box 648**
**Louisville, KY 40201, USA**
(C.O.D. and Foreign Orders - Add $5 Shipping/Handling)
References: Bank of Louisville, Citizens Fidelity Bank, Louisville Chamber of Commerce

VISA master charge

**CIRCLE 48 ON READER SERVICE CARD**

routines that you want to call into memory. Chaining permits *COMMON*ed variables but not *CHAIN/MERGE*. Error-handling is quite sophisticated, both local and global, and allows trapping of disk I/O errors. It is possible to trap errors within subroutines or defined functions and retain normal execution or to exit loops prematurely.

BASIC/Z handles three sorts of files: sequential, CP/M random files, and its own version of random files, in which the first record is used to store the number of records and logical record length. Record packing is done in a variety of ways: numeric, string, or array. Record locking is offered in the MP/M and TurboDos environments.

The language doesn't use *LPRINT*, *PRINT DEV#*, etc., to control output but rather one print stream which can be redirected by *CONSOLE*, *LPRINTER*, and *SPOOLA* statements. This makes it possible to a certain extent to combine output routines. The record and file-handling commands are unique to BASIC/Z and not similar to MS-BASIC commands. Formatted numeric printing is completely programmer definable. Input string fields can be displayed with preset lengths, fill characters, and offer the user a variety of editing options.

BASIC/Z is an attractive improvement on generic CP/M MS-BASIC. While it doesn't have the flexibility and versatile programming environment of Alcor's Multi-BASIC, it is a good package with its own strengths. The manual is thorough and comprehensive, and installation is easy. BASIC/Z performed well on the benchmarks, though not as well as Multi-BASIC. Like Multi-BASIC, the other compiler we reviewed in CP/M BASICs, BASIC/Z did poorly on the trig tests. Again, as with Multi-BASIC, the .COM files were large, as the added run-time package is about 20K.

*The* **C Interpreter:**
***Instant-C***™

Programming in C has never been Faster.
Learning C will never be Easier.

***Instant-C***™ is an optimizing interpreter for the C language that can make programming in C three or more times faster than using old-fashioned compilers and loaders. The interpreter environment makes C as easy to use and learn as Basic. Yet ***Instant-C***™ is 20 to 50 times faster than interpreted Basic. This new interactive development environment gives you:

**Instant Editing.** The full-screen editor is built into ***Instant-C***™ for immediate use. You don't wait for a separate editor program to start up.

**Instant Error Correction.** You can check syntax in the editor. Each error message is displayed on the screen with the cursor set to the trouble spot, ready for your correction. Errors are reported clearly, by the editor, and only one at a time.

**Instant Execution.** ***Instant-C***™ uses no assembler or loader. You can execute your program as soon as you finish editing.

**Instant Testing.** You can immediately execute any C statement or function, set variables, or evaluate expressions. Your results are displayed automatically.

**Instant Symbolic Debugging.** Watch execution by single statement stepping. Debugging features are built-in; you don't need to recompile or reload using special options.

**Instant Loading.** Directly generates .EXE or .CMD files at your request to create stand-alone versions of your programs.

**Instant Floating Point.** Uses 8087* co-processor if present.

**Instant Compatibility.** Follows K & R standards. Comprehensive standard library provided, with source code.

**Instant Satisfaction.** Get more done, faster, with better results. ***Instant-C***™ is available now, and works under PC-DOS, MS-DOS*, and CP/M-86*. Money back guarantee within 30 days.

Find out how ***Instant-C***™ is changing the way that programming is done. ***Instant-C***™ is $495. Call or write for more information.

Rational
Systems, Inc.

(617) 653-6194
P.O. Box 480
Natick, Mass. 01760

Trademarks: MS-DOS (Microsoft Corp.), 8087 (Intel Corp.), CP/M-86 (Digital Research, Inc.), Instant-C (Rational Systems, Inc.)

CIRCLE 72 ON READER SERVICE CARD

## MACINTOSH BASICS

### Apple MacBASIC

Like most of the Macintosh applications, Apple Computers Inc.'s MacBASIC uses windows to edit program source code. The mouse can be used for selecting text, and standard cut, copy, and paste commands are available. What's more, several windows can be opened with different files available so the programmer can cut and copy routines between different programs easily.

And as if this wasn't enough, MacBASIC is multitasking, allowing several programs to be running simultaneously with each window being updated on the fly. Naturally this means that each program will be proportionally slower, but for many applications speed will not be critical.

Variable names can be any length, and all characters are significant. Data types include: Boolean, extended real, double real, single real, integer, long integer, character and string. The extended precision real is an 80-bit numeric type for really large number crunching.

Interestingly enough, the documentation said that double- and single-precision reals were actually slower than the extended type since all were converted to extended for calculations. As shown in the table, strings may be up to 65,535 characters long. Strings may be delimited by either single or double quotes. This allows either quotation mark to be placed inside the string itself.

MacBASIC allows several advanced forms of flow control including *CASE/SELECT* and *DO/LOOP*. All forms of flow control can be terminated by an *EXIT* command. *IF-THEN-ELSE* structures can be on multiple lines, and the *EXIT* command can appear at any point.

Three types of subroutines are available: functions, subs and programs. Functions come in two varieties. In the first the standard *DEF* statement precedes the definition of a single line function. The prefix *FN* is not required with MacBASIC functions, thus giving greater naming flexibility.

The second function type can be multiline and allows the programmer to specify which type of data will be returned by the function. MacBASIC functions can also be recursive. Subroutines can have parameter passing, and parameters

# ADVANTAGE #7

At Programmer's Connection we are fighting to give you the best prices on software development tools. We regularly offer significant discounts and savings on most of our products. Value conscious programmers will find our specials on featured items are right on target. So take aim at savings; call Programmer's Connection the pioneers in software value.

**Discover the advantages of buying from Programmer's Connection:**

1. We offer the latest version of a product.
2. Most popular products are in stock ready to be shipped.
3. Receive same manufacturer's support as if buying direct.
4. Experienced professional programmers are on staff.
5. Choose from a large selection of the best software products available.
6. Knowledgeable and courteous sales staff.
7. Significant discounts off of retail prices.
8. No extra charge on prepaid orders, including major credit cards.
9. Reasonable charges for shipping and handling.
10. Toll free services from Canada and the U.S.

## C LANGUAGE:

| | List | Ours |
|---|---|---|
| **Computer Innovations C-86 Compiler** | 395 | **299** |
| **DeSmet C Compiler with Debugger** | 159 | **145** |
| **Lattice C Compiler** from Lattice *(Recommended!)* | 500 | **339** |
| **Lattice C Compiler** from Lifeboat | 500 | **299** |
| **Mac C** by Consulair *for MacIntosh* | 295 | **259** |
| **Xenix Development System** by SCO | 1350 | **1099** |

### Mark Williams C Compiler with Source Debugger

This highly optimized C compiler includes an extremely useful source-level debugger that can save you hours of debugging time. A truly professional C development system.

List Price $495 Our Price **$429**

## OTHER LANGUAGES:

| | List | Ours |
|---|---|---|
| **8088 Assembler w/Z-80 Translator** · 2500 AD | 100 | **89** |
| **APL*Plus/PC** by STSC | 595 | **469** |
| **BetterBASIC** by Summit Software | 200 | **169** |
| **Golden Common LISP** by Gold Hill | 495 | **439** |
| **Macro Assembler** by Microsoft | 150 | **119** |
| **Modula-2/86** by Logitech | 495 | **439** |
| **Professional BASIC** by Morgan Computing | 99 | **89** |
| **Turbo Pascal 3.0 w/BCD & 8087** by Borland | 125 | **109** |

## C UTILITIES:

| | List | Ours |
|---|---|---|
| **C Power Paks** from Software Horizons | Call | **Call** |
| **C-Sprite** *Symbolic Debugger for Lattice C* | 175 | **159** |
| **c-tree** by FairCom | 395 | **359** |
| **C Utility Library** by Essential Software | 149 | **119** |
| **dBC** *dBase/C Interface* by Lattice | 250 | **219** |
| **DOS LINK** *Support for DeSmet C* | 35 | **35** |
| **English-to-C/C-to-English** by Catalytix | 100 | **100** |
| **ESP for C** by Bellesoft | 349 | **279** |
| **Graphic C** by Scientific Endeavors | 250 | **209** |
| **Greenleaf C Functions Library** | 185 | **139** |
| **Greenleaf Comm Library** | 185 | **139** |
| **Halo** *Graphics* by Media Cybernetics | 200 | **125** |
| **PANEL** *Screen Designer* by Roundhill | 295 | **234** |
| **Run/C Interpreter** by Age of Reason | 150 | **129** |

### Pre-C by Phoenix Software

A complete lint-like utility that helps detect logic errors by searching for inconsistencies in functions and data types across multiple files. A powerful addition to anyone's C toolbox.

List Price $395 Our Price **$339**

## C UTILITIES:

| | List | Ours |
|---|---|---|
| **Safe C Standalone Interpreter** by Catalytix | 400 | **400** |
| **Safe C Dynamic Profiler** by Catalytix | 150 | **150** |
| **Safe C Runtime Analyzer** by Catalytix | 400 | **400** |
| **Windows For C** by Creative Solutions | 195 | **139** |

## OTHER PRODUCTS:

| | List | Ours |
|---|---|---|
| **APL2C** by Decision Images *Interfaces APL to C* | 150 | **139** |
| **Btrieve** by SoftCraft | 250 | **199** |
| **Codesmith-86 Debugger** by Visual Age | 145 | **129** |
| **Dr. Halo** by Media Cybernetics | 95 | **79** |
| **FORTRAN Libraries** by Alpha Comp. Serv. | Call | **Call** |
| **FORTRAN Scientific Subroutine Library** | 175 | **159** |
| **Periscope** *Debugger* by Data Base Decisions | 295 | **269** |
| **Pfix-86 Plus** by Phoenix | 395 | **299** |
| **Plink-86** *Overlay Linker* by Phoenix | 395 | **299** |
| **Pmate** *Macro Text Editor* by Phoenix | 225 | **159** |
| **Polytron Products** *We Carry a Full Line* | Call | **Call** |
| **Screen Sculptor** by Software Bottling | 125 | **109** |
| **XTC** *Text Editor* by Wendin | 99 | **89** |
| **Xtrieve** by SoftCraft *Sale!* | 195 | **149** |

### EPSILON Macro Text Editor

This extremely fast and productive programmer's editor features the ability to run a compiler or other program in one window while editing files in other windows. It has powerful EMACS editing commands, multiple windows, error recovery and much more.

List Price $195 Our Price **$179**

### The Profiler by DWB Associates

This valuable program optimization tool allows you to isolate the most often executed parts of your programs. It works with most languages and compilers.

List Price $125 Sale Price **$69**

Prices are subject to change without notice.
Account is charged when order is shipped.

VISA MasterCard

**1-800-336-1166**

In Canada: **1-800-225-1166**

**Call for our new Spring Catalog**

PC

**Programmer's Connection**
136 Sunnyside Street
Hartville, Ohio 44632
(216) 877-3781 (In Ohio)

**"Programmers Serving Programmers"**

CIRCLE 23 ON READER SERVICE CARD

## Functions, subroutines, and overlays

| Manufacturer and product | One line | Multiline | Recursive | Local variables | External | Call by line number | Call by label | Takes arguments | Local variables | External | Use COMMON to pass data | Allow CHAIN MERGE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **MS-DOS interpreters** | | | | | | | | | | | | |
| American Planning Corp. MEGABASIC | yes | yes | yes | yes | yes | yes | yes | yes | yes | yes | yes | yes |
| Control-C BASIC Interpreter | yes | no | no | no | no | yes | yes[1] | yes | yes | yes | no | no |
| IBM/Microsoft BASICA | yes | no | no | no | no | yes | no | no | no | no | yes | yes |
| Microsoft MS-BASIC 5.28 | yes | no | no | no | no | yes | no | no | no | no | yes | yes |
| Morgan Computing Professional BASIC | yes | no | no | no | no | yes | yes | no | no | no | yes | yes |
| Ryan-McFarland RM-BASIC | yes | yes | no | no | no | yes | yes | no | no | no | yes[2] | no |
| Southwest Data Pluto BASIC | yes | no | no | no | no | yes | yes[1] | yes | yes | yes | no | no |
| Summit Software Better BASIC | yes | yes | yes | yes | yes | yes | yes | yes | yes | yes | no | no |
| TransERA TBASIC | yes | no | no | no | no | yes | yes | yes | yes | yes | no | no |
| True BASIC | yes | yes | yes | yes | yes | yes | yes | yes | yes | yes | no | no |
| WATCOM BASIC | yes | yes | yes | yes | no | yes | yes | yes | yes | no | yes | no |
| **MS-DOS compilers** | | | | | | | | | | | | |
| Digital Research CBASIC | yes | yes | no | yes | yes | yes | yes | yes[2] | yes[2] | yes[2] | yes | no |
| IBM/Microsoft IBM BASIC Compiler | yes | no | no | no | no | yes | yes | no | no | no | yes | no |
| Microsoft Business BASIC Compiler | yes | yes | no | no | no | yes | yes | no | no | yes | yes | no |
| Microsoft BASIC Compiler | yes | no | no | no | no | yes | yes | no | no | no | yes | no |
| MicroWay 87BASIC | yes | no | no | no | no | yes | yes | no | no | no | yes | no |
| Softaid MTBASIC | no | no | no | no | no | yes | no | no | no | no | no | no |
| Sparry BASIC B | no | no | no | no | no | yes | no | no | no | no | yes | no |
| **CP/M 86 interpreter** | | | | | | | | | | | | |
| Digital Research Personal BASIC | yes | no | no | no | no | yes | no | no | no | no | yes | yes |
| **CP/M interpreters** | | | | | | | | | | | | |
| DeltaSoft DeltaBASIC | yes | no | no | no | no | yes | no | no | no | no | yes | no |
| Micro Mike's BaZic | yes | yes | no | yes | no | yes | no | no | no | no | no | no |
| Spectrum Logic DBASIC | yes | no | no | no | no | yes | no | no | no | no | no | NC |
| **CP/M compilers** | | | | | | | | | | | | |
| Alcor Systems Multi-BASIC | yes | yes | yes | yes | yes | yes | yes | yes | yes | yes | no | no |
| System/z BASIC/Z | yes | yes | yes | yes | no | yes | yes | no | no | no | yes | no |
| **Apple Macintosh** | | | | | | | | | | | | |
| Apple MacBASIC | yes | yes | yes | yes | no | no | yes | yes | yes | no | yes | yes |
| Microsoft MS-BASIC 1.0 | yes | no | no | no | no | yes | no | no | no | no | yes | yes |
| Microsoft MS-BASIC 2.0b | yes | yes | no | yes | no | yes | yes | yes | yes | no | yes | yes |
| Microsoft MS-BASIC 2.0d | yes | yes | no | yes | no | yes | yes | yes | yes | no | yes | yes |

NC=Not Chained.

[1] Called as Public Programs. [2] USE clause is employed instead of COMMON.

Table 4.

become local variables in the subroutine, thus having no effect on the variables in the main program.

To use programs as subroutines, the *PERFORM* command is used. When one program calls another program, any number of variables can be passed between them. Upon completion of the called program, the caller is restarted at the point where the *PERFORM* statement occurred, and all of the original variable values are retained.

MacBASIC has two interrupt handlers: KBD and ERROR. KBD monitors the keyboard for any key press and forces the program to jump to a user-defined routine which handles the keyboard event. ERROR interrupts can be used for standard error handling like *ON ERROR GOTO*, used in other versions of BASIC.

The Macintosh is best known for its graphics capabilities. MacBASIC makes use of most of the Quickdraw routines residing in the Mac's ROM. Also, windows can be independently manipulated, and graphics can be output to the printer as well as the screen. Data and graphics can be shared with other Macintosh applications through the use of the Clipboard.

As shown in the tables, MacBASIC can be up to four times faster than MS-BASIC on the Macintosh. This along with its many advanced features should make MacBASIC a contender despite its late entry into the field.

### Microsoft MS-BASIC

When the Macintosh was first introduced a year ago, the only language that was publicly available was MS-BASIC from Microsoft Corp. Because of this, a good many application programs have been written for the Mac with this language.

Unfortunately, the original version was little more than a ported standard that made little use of the Mac's special environment. None the less, for a year later there were no competitors in the BASIC market for the Mac, and Microsoft has now released a second version that takes care of most of the original's shortcomings.

With version 2.0 of MS-BASIC, lots more of the Mac's built-in graphics and user interface routines have been added to the interpreter. Also, those ROM routines that are used have better documentation than version 1.0 gave. We have included both versions in the table for comparison.

Note that version 2.0 itself comes in two separate versions: (b) and (d). These denote binary and decimal math, respectively. The former uses BCD numbers which do floating point calculations faster. The latter uses IEEE 64-bit, floating point numbers for greater accuracy. Practically speaking, the decimal math version is equivalent to version 1.0, and most of the programs which ran under that interpreter should run under the version 2.0 (d) interpreter. Both 2.0 (b) and (d) versions are included on the same disk.

All of these MS-BASIC versions share a number of things in common. Functions must have single-line definitions, and recursion is not possible. The *FN* prefix is necessary. Data types are double-precision real, single-precision real, integers and strings. There are no provisions for Booleans or user-defined types. The documentation for version 1.0 said that arrays could not be dynamically allocated, but I discovered through experimentation that this was not true. Obviously, this version was rushed to market.

Version 1.0 was a standard MS-BASIC package, and I found that most programs that ran on other machines under MS-BASIC would run on the Macintosh with little or no modification. Version 2.0 is a completely revised BASIC and has the following improvements:

Line numbers are no longer required. Labeled subroutines can be used. SUB programs are allowed which can have local variables and parameter passing. Macintosh ROM routines for Menus, Windows, Dialog boxes, and Buttons have been added. Besides *ON ERROR GOTO*, three other interrupts are handled: *ON MENU GOTO*, *ON TIMER GOTO*, and *ON DIALOG GOTO*.

The communications port can be set from within BASIC, instead of using a machine-language routine to do this.

GOOD NEWS!

C for the 6809 WAS NEVER BETTER!

**INTROL-C/6809,Version1.5**

Introl's highly acclaimed 6809 C compilers and cross-compilers are now more powerful than ever!

We've incorporated a totally new 6809 Relocating Assembler, Linker and Loader. Initializer support has been added, leaving only bitfield-type structure members and doubles lacking from a 100% full K&R implementation. The Runtime Library has been expanded and the Library Manager is even more versatile and convenient to use. Best of all, compiled code is just as compact and fast-executing as ever - and even a bit more so! A compatible macro assembler, as well as source for the full Runtime Library, are available as extra-cost options.

Resident compilers are available under **Uniflex, Flex** and **OS9.**

Cross-compilers are available for **PDP-11/UNIX** and **IBM PC/PC DOS** hosts.

Trademarks:
Introl-C, Introl Corporation
Flex and Uniflex, Technical Systems Consultants
OS9, Microware Systems
PDP-11, Digital Equipment Corp.
UNIX, Bell Laboratories
IBM PC, International Business Machines

For further information, please call or write.

INTROL
CORPORATION
647 W. Virginia St.
Milwaukee, WI 53204
(414) 276-2937

CIRCLE 32 ON READER SERVICE CARD

## File I/O and graphics

| Manufacturer and product | Formatted output (PRINT USING) | Random access I/O | Pack numeric data into strings | Port I/O | Graphics | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | Hi Res | Window | Viewport | Allow user scale | Has clipping |
| **MS-DOS interpreters** | | | | | | | | | |
| American Planning Corp. MEGABASIC | yes | yes | yes | yes | yes | no | no | no | no |
| Control-C BASIC Interpreter | yes | yes | no | yes | no | no | no | no | no |
| IBM/Microsoft BASICA | yes | yes | yes | yes | yes | yes | yes | yes | yes |
| Microsoft MS-BASIC 5.28 | yes | yes | yes | yes | no | no | no | no | no |
| Morgan Computing Professional BASIC | yes | yes | yes | yes | yes | no | no | no | no |
| Ryan-McFarland RM-BASIC | yes | yes | no | yes | yes | no | no | no | no |
| Southwest Data Pluto BASIC | yes | yes | no | no | no | no | no | no | no |
| Summit Software Better BASIC | yes | yes | no | yes | yes | yes | no | yes | yes |
| TransERA TBASIC | yes | yes | yes | yes | yes | yes | yes | yes | yes |
| True BASIC | yes | yes | yes | no | yes | yes | yes | yes | yes |
| WATCOM BASIC | yes | yes | yes | yes | yes | no | no | no | no |
| **MS-DOS compilers** | | | | | | | | | |
| Digital Research CBASIC | yes | yes | no | yes | yes | yes | yes | yes | yes |
| IBM/Microsoft IBM BASIC Compiler | yes | yes | yes | yes | yes | no | no | no | no |
| Microsoft Business BASIC Compiler | yes | yes | yes | yes | no | no | no | no | no |
| Microsoft BASIC Compiler | yes | yes | yes | yes | no | no | no | no | no |
| MicroWay 87BASIC | yes | yes | yes | yes | yes | no | no | no | no |
| Softaid MTBASIC | no | yes | yes | yes | no | no | no | no | no |
| Sparry BASIC B | no | yes | no | yes | no | no | no | no | no |
| **CP/M 86 interpreter** | | | | | | | | | |
| Digital Research Personal BASIC | yes | yes | yes | yes | no | no | no | no | no |
| **CP/M interpreters** | | | | | | | | | |
| DeltaSoft DeltaBASIC | yes | yes | yes | no | no | no | no | no | no |
| Micro Mike's BaZic | yes | yes | no | no | no | no | no | no | no |
| Spectrum Logic DBASIC | yes | yes | yes | yes | no | no | no | no | no |
| **CP/M compilers** | | | | | | | | | |
| Alcor Systems Multi-BASIC | yes | yes | yes | yes | no | no | no | no | no |
| System/z BASIC/Z | yes | yes | yes | yes | no | no | no | no | no |
| **Apple Macintosh** | | | | | | | | | |
| Apple MacBASIC | yes | yes | yes | yes | yes | yes | yes | yes | yes |
| Microsoft MS-BASIC 1.0 | yes | yes | yes | yes | yes | yes | yes | yes | yes |
| Microsoft MS-BASIC 2.0b | yes | yes | yes | yes | yes | yes | yes | yes | yes |
| Microsoft MS-BASIC 2.0d | yes | yes | yes | yes | yes | yes | yes | yes | yes |

Table 5.

# ORDER COMPUTER LANGUAGE BACK ISSUES WHILE THEY LAST!

*Complete your collection of COMPUTER LANGUAGE magazines with our selection of back issues. A complete set is sure to become a valuable collector's item in years to come. Here are just a few of the features included in each issue:*

## PREMIER

***The biggest collector's issue***

- Basic Becomes a Structured Language—by Kemeny & Kurtz
- Programming in the Unix Environment
- COBOL: Pride and Prejudice
- Exploring Ada and Modula-2

Exotic Language: SNOBOL

Interview: Charles Moore

## OCTOBER '84

- An Implementation Demonstrating C Portability
- The Evolution of ZCPR–Part I
- BATCH—A Powerful IBM "Language"

Exotic Language: PILOT

Interview: Donald Knuth

## NOVEMBER '84

- Enhancing Source Code Control under UNIX, Part 1
- Natural Language Processing and LISP
- Building Portable Programs

Exotic Language: OCCAM

Interview: Gary Kildall

## DECEMBER '84

- Exploratory Programming
- Fred: A Language within Framework
- Six Pascal Compilers Compared

Exotic Language: OMNI

Interview: Bill Godbout

## JANUARY '85

- Macros and Procedures
- Extensibility in Forth
- The IIIrd Dimension-Programming in dBASE III

Exotic Language: Transaction Application Language

Interview: Sol Libes

## FEBRUARY '85

***C Language Special Issue***

- Twenty-One C Compilers Compared
- The Standardization of C
- C to Assembly Interface

Exotic Language: MUMPS

Interview: P.J. Plauger

*Only a limited quantity of magazines is available, so order today. To receive your back issues, just fill out this coupon and mail it back with a check for $4.00 per issue.*

**Premier** ________ copies × $4.00 = $ ________

**Oct. '84** ________ copies × $4.00 = $ ________

**Nov. '84** ________ copies × $4.00 = $ ________

**Dec. '84** ________ copies × $4.00 = $ ________

**Jan. '85** ________ copies × $4.00 = $ ________

**Feb. '85** ________ copies × $4.00 = $ ________

**Mar. '85** ________ copies × $4.00 = $ ________

**Apr. '85** ________ copies × $4.00 = $ ________

Foreign orders: Add $3.00 for airmail. Total $________

NAME ________________

COMPANY ________________

ADDRESS ________________

CITY, STATE, ZIP ________________

5

**Send payment and coupon to:**

COMPUTER LANGUAGE

**Back Issues**
**131 Townsend St.**
**San Francisco, CA 94107**

Graphics output can be directed to the printer in the same way it is sent to the screen. Also, different font styles can be sent to the printer now.

Where version 1.0 uses a single-line editor, 2.0 has a full-screen editor which includes a global find-and-replace feature. This is a big help for program development. All of these new features are welcome additions. What is missing, however, is the ability to compile the programs to get the full speed of the Mac's M68000 processor.

### Pterodactyl Macintosh BASIC

As of this writing we had also received a demonstration of a BASIC package for the Macintosh from Pterodactyl Software Inc. This package takes programs written in BASICA for the IBM PC and converts them to compiled applications on the Mac. The programmer can modify the source code to make use of many Mac features, then use the compiler to generate an application. When the object code is created, the compiler automatically creates a resource file which handles window display, menu selection, and other Mac options.

A unique aspect of the compiler is that it takes instructions utilizing the function keys on the IBM and converts them to Menu selections on the Macintosh. The programmer may use standard Macintosh resource editors to modify the finished product and fine-tune the display. Since the resource file is maintained and compiled separately from the BASICA file, the graphics tuning can take place without affecting the main program code.

Other nice features of the Pterodactyl compiler are automatic window updating and extensions that allow calling Mac toolbox procedures and functions. Also, desk accessories are handled by the software, and groups of buttons in dialog boxes are automatically toggled,

The release date for this product is expected to be in the summer of 1985. It will require a 512K Macintosh with an external microfloppy drive or hard disk. The Macintosh 68000 Assembler Development System will also be needed. Another version to be used on the Lisa (Mac XL) is expected out by the time you read this.

| Manufacturer | Product |
|---|---|
| American Planning Corp.<br>4600 Duke St. Ste. 423<br>Alexandria, Va. 22304<br>(800) 368-2248 | MEGABASIC |
| Addison-Wesley Publishing Co. Inc.<br>Jacob Way<br>Reading, Mass. 01867<br>(617) 944-6795 | True BASIC |
| Digital Research Inc.<br>P.O. Box 579<br>160 Central Ave.<br>Pacific Grove, Calif. 93950<br>(408) 649-3896 | CBASIC,<br>Personal BASIC |
| Sparry Software Labs<br>Box 632<br>16 Park Lane Ave.<br>Milford, Mass. 01757<br>(617) 473-5435 | BASIC B |
| Microsoft Corp.<br>10700 Northup Way<br>Box 97200<br>Bellevue, Wash. 98009<br>(206) 828-8080 | Business BASIC Compiler,<br>BASIC Compiler,<br>MS-BASIC 5.28,<br>MS-BASIC 1.0,<br>MS-BASIC 2.0b,<br>MS-BASIC 2.0d |
| MicroWay<br>P.O. Box 79<br>Kingston, Mass. 02364 | 87BASIC |
| Control-C Software Inc.<br>6441 S. W. Canyon Ct.<br>Portland, Ore. 97221<br>(503) 292-8842 | BI-286 |
| Ryan-McFarland<br>609 Deep Valley Dr.<br>Rolling Hills Estates, Calif. 90274<br>(213) 541-4828 | RM-BASIC |
| WATCOM Products Inc.<br>415 Phillip St.<br>Waterloo, Ont. N2L 3X2<br>Canada | WATCOM BASIC |
| Southwest Data Systems Inc.<br>3017 San Fernando Blvd.<br>Burbank, Calif. 91504<br>(818) 841-1610 | Pluto BASIC |
| Morgan Computing Co. Inc.<br>P.O. Box 112730<br>Dallas, Texas 75011<br>(214) 245-4763 | Professional BASIC |
| Summit Software Technology Inc.<br>40 Grove St.<br>Wellesley, Mass. 02181<br>(617) 235-0729 | Better BASIC |
| Softaid Inc.<br>Box 2412<br>Columbia, Md. 21045<br>(301) 792-8096 | MTBASIC |
| System/z Inc.<br>P. O. Box 11<br>Richton Park, Ill. 60471<br>(312) 481-8085 | BASIC/Z |
| Alcor Systems<br>1132 Commerce<br>Richardson, Texas 75081<br>(214) 238-8554 | Multi-BASIC |
| Micro Mike's Inc.<br>3015 Plains Blvd.<br>Amarillo, Texas 79102<br>(806) 372-3633 | BaZic |
| DeltaSoft Inc.<br>P. O. Box 2029<br>Tyler, Texas 75710<br>(214) 581-1425 | DeltaBASIC |
| Spectrum Logic Corp.<br>P. O. Box 41006<br>Nashville, Tenn. 37204-1006<br>(615) 683-2903 | DBASIC |
| Pterodactyl Software, Inc.<br>200 Bolinas Rd. #27<br>Fairfax, Calif. 94930<br>(415) 485-0714 | Pterodactyl BASIC |

STRUCTURE FOR BASIC

With PROFESSIONAL PROGRAMMING ENVIRONMENT

ELIMINATES LINE NUMBERS

ALLOWS MULTI-LINE CONDITIONALS

PCDOS/MSDOS

BENDORF ASSOCIATES
6006 S. MAIN
P.O. BOX 5910
ROSWELL, NM
88201
505 347-5701

VISA/MASTERCARD

WORKS WITH BASICA INTERPRETER COMPILER

FULL ERROR LOGGING

PROGRAM LISTER

LABELED PROCEDURES

MACROS

SUB-ROUTINES

LIBRARIES

$49.95

CIRCLE 6 ON READER SERVICE CARD

**Use our imagination.**

Get in touch with the high tech ad agency that speaks your language. Tap into our long experience with trade and technical accounts.

You'll find just the right mix of solid marketing savvy and fresh creative concepts.

**Call (415) 827-4111.**

**Steve Rank Inc.**
*1260 Monument Boulevard*
*Concord, California 94520*

CIRCLE 92 ON READER SERVICE CARD

Hyperon Software

Specializing in innovative programming tools.

- Complete documentation and C-source provided (presently DOS only).
- Reasonable prices.
- High quality and good performance.

*Products currently available:*

C Preprocessor. Features include variables and expressions, loops, and full macros. Price — $39.95.

General purpose editor. Line oriented commands with a screen oriented submode. Command window. Price — $29.95

*Order from:*

**HYPERON SOFTWARE**
**P.O. Box 3349**
**Costa Mesa, CA 92628**

Enclose check or money order. California residents add 6%.
2532 Orange Ave., Costa Mesa, CA

CIRCLE 51 ON READER SERVICE CARD

NEW!

**Advanced Trace86™**

*Symbolic Debugger & Assembler Combo*

- Full-screen trace with single stepping; Even backstepping!
- Write & Edit COM & EXE programs
- Conditional breakpoints (programmable)
- Switch between trace and output screen; Or set up two monitors
- 8087, 80186, 80286, 80287 support
- Write labels & comments on code
- Polish hex/decimal calculator
- and more . . . Priced at $175.00

*To order or request more information contact:*

**Morgan Computing Co., Inc.**
800/527-0009
P.O. Box 112730, Dallas, TX 75011
(214) 739-5895

CIRCLE 22 ON READER SERVICE CARD

**PC BASIC PROGRAMMER'S UTILITIES**
**TRACE—COMPILE—COMPARE**

AUTOTRACE powerful single-step or continuous trace in COMPILED or interpreter BASIC. BREAK on variables or line numbers. Change values at any time. RECALL screen displays—save text that scrolls off your screen. Full 80-column printout saves paper; only CHANGED variable values printed. Save trace to disk. This utility finds your logic errors fast!!

AUTOCPL precompiler system. Automates compile/link process. No more manual editing to remove or change code for compilation. No more separate versions to save. Include SLASH/N to remove line numbers. Now compiling is easy, fast, automatic!!

AUTOCOMPARE fast compare of two programs or ASCII files. Prints 132-column or 80-column record of all differences, or save to disk. Includes utility to compare WORDSTAR (R) files. NEW! will unformat indented ASCII Files to allow easy editing with WORDSTAR. Save hours of manual reformatting.

$49.95 each (each includes RAMdisk & SPOOLER)
All 3 for $99.95; S/H $2.50 ea.

TIMESHARE ASSOCIATES, INC. Dept. L
10202 Robinson
Overland Park, KS 66212
(913) 642-7564

CIRCLE 80 ON READER SERVICE CARD

**OPT-TECH SORT™**

**SORT/MERGE program for IBM-PC, XT & AT**

**Now also sorts dBASE II files!**

- Written in assembly language for **high performance** Example: 4,000 records of 128 bytes sorted to give key & pointer file in 30 seconds. **COMPARE!**
- Sort ascending or descending on up to nine fields
- Ten input files may be sorted or merged at one time
- Supports many file structures & data types
- Filesize limited only by your disk space
- Output file can be full records, keys or pointers
- Can be run from keyboard or as a batch command
- Can be called as a subroutine to many languages
- Easy to use — Fully documented
- **$99** — VISA, M/C, Check, Money Order, COD, or PO Quantity discounts and OEM licensing available

To order or to receive additional information write or call:

**OPT-TECH DATA PROCESSING**
P.O. Box 2167 Humble, Texas 77347
(713) 454-7428

Requires DOS, 64K and One Disk Drive

CIRCLE 62 ON READER SERVICE CARD

**FoxBASE™**
**Interpreter/Compiler**

- **dBASE II® source compatible**
- Runs 3-20 times faster than dBASE II
- 8087 coprocessor support
- 14 digit precision
- Up to 48 fields per record
- Full type-ahead capabilities
- Provides compact object code and program security
- Twice as many memory variables as dBASE II

**FOX SOFTWARE INC.**
13330 Bishop Road, P.O. Box 269
Bowling Green, OH 43402
419-354-3981

CIRCLE 29 ON READER SERVICE CARD

**YOU CAN C**

**VIEW v3**
The ultimate CP/M disk utility.

- **Edits** any disk sector
- **Displays** allocation bit map
- **Rebuilds** files automatically from selected blocks

If you ever had to recover a crashed disk, you need **VIEW**. Use **VIEW** to patch damaged sectors, un-erase files, examine other disk formats, more

**COMPLETE C SOURCE CODE $59**

**BD Software's C Compiler**
The most popular CP/M C Compiler comes with **symbolic debugger** and special Western Wares' **ISIS-II** function library **$140**

**C-PACK**
A disk full of useful CP/M utilities written in C, PEPE, DEL, BACKUP, DPATCH, ECHO, CHX, PAUSE, more Source incl **$19**

**Other C Products**
EZZAP, ICX, MEDIT, more
Send for a catalog today

CP/M®, Digital Research
ISIS-II ®, Intel Corp

Western Wares 303-327-4898
Box C • Norwood, CO 81423

CIRCLE 41 ON READER SERVICE CARD

**Scroll & Recall™**

***Screen and Keyboard Enhancement***
*for the IBM-PC XT AT & Compatibles*

*Allows you to conveniently scroll back through data that has gone off the top of your display screen. Up to 27 pages of data can be recalled or written to a disk file.*

*Allows you to recall, edit and re-enter your previously entered DOS commands and data lines, without retyping.*

*Very easy to use, fully documented. Compatible with all versions of DOS, monochrome & graphic displays.*

***$69** - Visa, M/C, Check, COD, POs*

***Make Your Work Easier!***

To Order or to Receive Additional Information, Write or Call:
**Opt-Tech Data Processing**
P.O. Box 2167 • Humble, Texas 77347
(713) 454-7428
*Dealer Inquires Welcome*

CIRCLE 64 ON READER SERVICE CARD

# ADVERTISER INDEX

| | PAGE NO. | CIRCLE NO. |
|---|---|---|
| Addison-Wesley Publishing Co. | 8 | 7 |
| Amber Systems | 65 | 2 |
| Atlantis Publishing Corp. | 78 | 3 |
| Awareco | 81 | 1 |
| BD Software | 84 | 4 |
| Bendorf & Associates | 95 | 6 |
| Blaise Computing Inc. | 60 | 8 |
| Borland International | cover IV | 9 |
| C Journal (The) | 83 | 5 |
| C Systems | 70 | 16 |
| C Ware | 82 | 11 |
| CCA Uniworks Inc. | 1 | 61 |
| Computer Helper Industries | 83 | 10 |
| COMPUTER LANGUAGE C Seminar | 69 | 97 |
| Creative Solutions | 85 | 14 |
| DWB Associates | 73 | 21 |
| Data Base Decisions | 23 | 30 |
| David Smith Software | 75 | 36 |
| Ecosoft | 59 | 17 |
| Entelekon | 51 | 50 |
| Essential Software Inc. | 86 | 33 |
| FairCom | 81 | 34 |
| Fox Software | 95 | 29 |
| General Communications | 56 | 60 |
| Gimpel Software | 35 | – |
| Greenleaf Software Inc. | 52 | 44 |
| HSC Inc. | 73 | 12 |
| Harvard Softworks | 74 | 47 |
| Human Computing Resources Corp. | 36 | 93 |
| Hyperon Software | 95 | 51 |
| Information Systems Inc. | 56 | 20 |
| Interface Technologies Corp. | 24 | 56 |
| Introl Corp. | 91 | 32 |
| JMI Software Consultants Inc. | 8 | 65 |
| KC Systems | 83 | 45 |
| Laboratory Microsystems Inc. | 81 | 35 |
| Lattice Inc. | 80 | 18 |
| Lifeboat Associates | 64 | 87 |
| Lifeboat Associates | 47 | 85 |
| mbp Software & Systems Technology | 53 | 53 |
| MEF Environmental | 61 | 15 |
| MEF Environmental | 87 | 55 |
| MIX Software | 41 | 42 |
| Manx Software Systems | cover III | 69 |
| Megamax Inc. | 83 | 13 |
| Micro Methods | 79 | 52 |
| Micro-Software Developers | 87 | 54 |
| Microsoft Corp. | 6 | 43 |
| Mindbank Inc. | 62 | 63 |
| Morgan Computing Co. | 95 | 22 |
| Mystic Canyon Software | 22 | 57 |
| Nantucket | 2 | 59 |
| Oasys | 77 | 84 |
| Oasys | 79 | 75 |
| Op-Tech Data Processing | 95 | 62 |
| Op-Tech Data Processing | 95 | 64 |
| Phoenix Computer Products Corp. | 67 | 24 |
| Poor Person Software | 71 | 67 |
| Programmer's Connection | 89 | 23 |
| Programmer's Shop (The) | 54 | 40 |
| RR Software Inc. | cover II | 58 |
| Rational Systems Inc. | 88 | 72 |
| Realia Inc. | 76 | 76 |
| SLR Systems | 77 | 73 |
| Shaw American Technologies | 87 | 48 |
| Soft Craft Inc. | 58 | 38 |
| Softaid Inc. | 79 | 78 |
| Software Horizons | 75 | 25 |
| Solution Systems | 63 | 37 |
| Solution Systems | 63 | 88 |
| Solution Systems | 63 | 89 |
| Solution Systems | 63 | 90 |
| Southwest Data Systems | 46 | 46 |
| Sparry Software Labs | 46 | 66 |
| Springer Verlag | 19 | 70 |
| Spruce Technology Corp. | 11 | 26 |
| Steve Rank Inc. | 95 | 92 |
| Summit Software Technology | 10 | 71 |
| SuperSoft | 12 | 74 |
| SuperSoft | 71 | 77 |
| SuperSoft | 59 | 79 |
| Systems Guild | 17 | 27 |
| Systems Peripheral Consultants | 75 | 91 |
| Timeshare Associates Inc. | 95 | 80 |
| TransEra Corp. | 42 | 82 |
| UniPress | 17 | 81 |
| Watcom Products Inc. | 4 | 28 |
| Western Wares | 95 | 41 |
| Whitesmiths Ltd. | 48 | 39 |
| Wizard Systems | 71 | 86 |
| Workman & Associates | 87 | 68 |
| Zeducomp | 73 | 83 |

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.

ADVERTISE
in the August issue of
COMPUTER LANGUAGE
Special
Exotic Language issue
Includes a Forth product wrap-up

Reservation deadline: June 5th

Contact:
Carl Landau
COMPUTER LANGUAGE
131 Townsend St.
San Francisco, Calif. 94107
(415) 957-9353

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

## BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 27346 PHILADELPHIA, PA USA

POSTAGE WILL BE PAID BY

COMPUTER
LANGUAGE

P.O. BOX 11747
PHILADELPHIA, PA 19101

---

# SUBSCRIBE

COMPUTER
LANGUAGE

Subscribe to **COMPUTER LANGUAGE** today for only $24.95—over 30% savings off the single copy price.

☐ Yes, start my Subscription to COMPUTER LANGUAGE today. The cost is only $24.95 for 1 year (12 issues).

☐ I want to increase my savings even more—send me 2 years (24 issues) of COMPUTER LANGUAGE for only $39.95.

☐ Payment enclosed ☐ Bill me

Name ______________________________

Company ______________________________

Address ______________________________

City, State, Zip ______________________________

**Please allow 6–8 weeks for delivery of first issue. Foreign orders must be prepaid in U.S. funds. Canada orders $30.95 per year. Outside the U.S., $36.95/year for surface mail or $54.95/year for airmail.**

BIR5

---

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 27346 PHILADELPHIA, PA USA

POSTAGE WILL BE PAID BY

COMPUTER
LANGUAGE

P.O. BOX 11747
PHILADELPHIA, PA 19101

---

SUBSCRIBE

COMPUTER
LANGUAGE

Subscribe to COMPUTER LANGUAGE today for only $24.95—over 30% savings off the single copy price.

☐ Yes, start my Subscription to COMPUTER LANGUAGE today. The cost is only $24.95 for 1 year (12 issues).

☐ I want to increase my savings even more—send me 2 years (24 issues) of COMPUTER LANGUAGE for only $39.95.

☐ Payment enclosed ☐ Bill me

Name ______________________________

Company ______________________________

Address ______________________________

City, State, Zip ______________________________

Please allow 6–8 weeks for delivery of first issue. Foreign orders must be prepaid in U.S. funds. Canada orders $30.95 per year. Outside the U.S., $36.95/year for surface mail or $54.95/year for airmail.

BIR5

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

# BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 22481 SAN FRANCISCO, CA USA

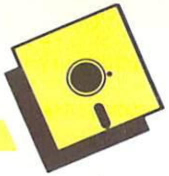
POSTAGE WILL BE PAID BY

**COMPUTER LANGUAGE**

2443 FILLMORE STREET • SUITE 346
SAN FRANCISCO, CA 94115

---

# FREE INFORMATION



Name ______________________

Company ______________________

Address ______________________

City, State, Zip ______________________

Country ____________ Telephone number ______________________

May issue. Not good if mailed after September 30, 1985.

**Circle numbers for which you desire information.**

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 11 | 21 | 31 | 41 | 51 | 61 | 71 | 81 | 91 | 101 | 111 | 121 | 131 | 141 |
| 2 | 12 | 22 | 32 | 42 | 52 | 62 | 72 | 82 | 92 | 102 | 112 | 122 | 132 | 142 |
| 3 | 13 | 23 | 33 | 43 | 53 | 63 | 73 | 83 | 93 | 103 | 113 | 123 | 133 | 143 |
| 4 | 14 | 24 | 34 | 44 | 54 | 64 | 74 | 84 | 94 | 104 | 114 | 124 | 134 | 144 |
| 5 | 15 | 25 | 35 | 45 | 55 | 65 | 75 | 85 | 95 | 105 | 115 | 125 | 135 | 145 |
| 6 | 16 | 26 | 36 | 46 | 56 | 66 | 76 | 86 | 96 | 106 | 116 | 126 | 136 | 146 |
| 7 | 17 | 27 | 37 | 47 | 57 | 67 | 77 | 87 | 97 | 107 | 117 | 127 | 137 | 147 |
| 8 | 18 | 28 | 38 | 48 | 58 | 68 | 78 | 88 | 98 | 108 | 118 | 128 | 138 | 148 |
| 9 | 19 | 29 | 39 | 49 | 59 | 69 | 79 | 89 | 99 | 109 | 119 | 129 | 139 | 149 |
| 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 | 110 | 120 | 130 | 140 | 150 |

I obtained this issue through:
- ☐ Subscription
- ☐ Computer Store
- ☐ Retail outlet
- ☐ Passed on by associate
- ☐ Other ____________

Comments ______________________

Attn: Reader Service Dept.

2/5

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 22481 SAN FRANCISCO, CA USA

POSTAGE WILL BE PAID BY

COMPUTER LANGUAGE

2443 FILLMORE STREET • SUITE 346
SAN FRANCISCO, CA 94115

FREE INFORMATION

Name ______________________

Company ______________________

Address ______________________

City, State, Zip ______________________

Country ____________ Telephone number ______________________

May issue. Not good if mailed after September 30, 1985.

Circle numbers for which you desire information.

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 11 | 21 | 31 | 41 | 51 | 61 | 71 | 81 | 91 | 101 | 111 | 121 | 131 | 141 |
| 2 | 12 | 22 | 32 | 42 | 52 | 62 | 72 | 82 | 92 | 102 | 112 | 122 | 132 | 142 |
| 3 | 13 | 23 | 33 | 43 | 53 | 63 | 73 | 83 | 93 | 103 | 113 | 123 | 133 | 143 |
| 4 | 14 | 24 | 34 | 44 | 54 | 64 | 74 | 84 | 94 | 104 | 114 | 124 | 134 | 144 |
| 5 | 15 | 25 | 35 | 45 | 55 | 65 | 75 | 85 | 95 | 105 | 115 | 125 | 135 | 145 |
| 6 | 16 | 26 | 36 | 46 | 56 | 66 | 76 | 86 | 96 | 106 | 116 | 126 | 136 | 146 |
| 7 | 17 | 27 | 37 | 47 | 57 | 67 | 77 | 87 | 97 | 107 | 117 | 127 | 137 | 147 |
| 8 | 18 | 28 | 38 | 48 | 58 | 68 | 78 | 88 | 98 | 108 | 118 | 128 | 138 | 148 |
| 9 | 19 | 29 | 39 | 49 | 59 | 69 | 79 | 89 | 99 | 109 | 119 | 129 | 139 | 149 |
| 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 | 110 | 120 | 130 | 140 | 150 |

I obtained this issue through:
- ☐ Subscription
- ☐ Computer Store
- ☐ Retail outlet
- ☐ Passed on by associate
- ☐ Other ____________

Comments ______________________

Attn: Reader Service Dept.

2/5

AZTEC

# The Most Powerful C

for the IBM AT • MACINTOSH • MS DOS • CP/M-80 • ROM APPLICATIONS
IBM PC/XT • APPLE // • CP/M-86 • TRSDOS • CROSS DEVELOPMENT

### Why Professionals Choose Aztec C

AZTEC C compilers generate fast, compact code. AZTEC C is a sophisticated development system with assemblers, debuggers, linkers, editors, utilities and extensive run time libraries. AZTEC C is documented in detail. AZTEC C is the most accurate and portable implementation of C for microcomputers. AZTEC C supports specialized professional needs such as cross development and ROM code development. MANX provides qualified technical support.

### AZTEC C86/PRO

**— for the IBM AT and PC/XT**

AZTEC C86/PRO provides the power, portability, and professional features you need to develop sophisticated software for PC DOS, MS DOS AND CP/M-86 based microsystems. The system also supports the generation of ROM based software for 8088/8086, 80186, and 80286 processors. Options exist to cross develop ROM code for 65xx, 8080, 8085, and Z80 processors. Cross development systems are also available that target most micro computers. Call for information on AZTEC C86/PRO support for XENIX and TOPVIEW.

**POWERFUL —** AZTEC C86/PRO 3.2 outperforms Lattice 2.1 on the DHRYSTONE benchmark 2 to 1 for speed (17.8 secs vs 37.1) while using 65% less memory (5.8k vs 14k). The AZTEC C86/PRO system also compiles in 10% to 60% less time and supports fast, high volume I/O.

**PORTABLE —** MANX Software Systems provides **real** portability with a family of compatible AZTEC C software development systems for PC DOS, MS DOS, CP/M-86, Macintosh, CP/M-80, APPLE //+, //e, and //c (NIBBLE - 4 apple rating), TRSDOS (80-MICRO - 5 star rating), and Commodore C64 (the C64 system is only available as a cross compiler - call for details). AZTEC C86/PRO is compatible with UNIX and XENIX.

**PROFESSIONAL —** For professional features AZTEC C86/PRO is unparalleled.

- Full C Compiler (8088/8086 - 80186 - 80286)
- Macro Assembler for 8088/8086/80186/80206
- Linkage Editor with ROM support and overlays
- Run Time Libraries - object libraries + source DOS 1.x; DOS 2.x; DOS 3.x; screen I/O; Graphics; UNIX I/O; STRING; simulated float; 8087 support; MATH; ROM; CP/M-86
- Selection of 8088/8086, 80186, or 80286 code generation to guarantee best choice for performance and compatibility
- Utility to convert AZTEC object code or libraries to Microsoft format. (Assembly + conversion takes less than half the time as Microsoft's MASM to produce MS object)
- Large memory models and sophisticated memory management
- Support products for graphics, DB, Screen, & ...
- ROMable code + ROM support + separate code and data + INTEL Hex Converter
- Symbolic Debugger & Other Utilities
- Full Screen Editor (like Vi)
- CROSS Compilers are available to APPLE //, Macintosh, CP/M-80, TRSDOS, COMMODORE C64, and ROM based 65xx, and 8080/8085/Z80
- Detailed Documentation

| | |
|---|---|
| AZTEC C86/PRO-AT<br>(configured for IBM AT - options for 8088/8086) | $500 |
| AZTEC C86/PRO-PC/XT<br>(configured for IBM PC/XT - options for 80186/80286) | $500 |

AZTEC C86/BAS includes C compiler (small model only), 8086 MACRO assembler, overlay linker, UNIX, MATH, SCREEN, and GRAPHICS libraries, debugger, and editor.

| | |
|---|---|
| AZTEC C86/BAS | $199 |
| AZTEC C86/BAS (CP/M-86) | $199 |
| AZTEC C86/BAS (DOS + CP/M-86) | $299 |
| UPGRADE to AZTEC C86/PRO | $310 |
| C-TREE Database with source | $399 |
| C-TREE Database (object) | $149 |

### CROSS COMPILERS

Cross Compilers for ROM, MS DOS, PC DOS, or CP/M-86 applications.

| | |
|---|---|
| VAX - > 8086/80xxx cross | $5000 |
| PDP-11 - > 8086/80xxx cross | $2000 |

Cross Compilers with PC DOS or CP/M-86 hosts are $750 for the first target and $500 for each additional target. Targets: 65xx; CP/M-80; C64; 8080/8085/Z80; Macintosh; TRSDOS; 8086/8088/80186/80286; APPLE //.

### AZTEC C68K

**— for the Macintosh**

For power, portability, and professional features AZTEC C68K-c is the finest C software development system available for the Macintosh.

The AZTEC C68K-c system includes a 68000 macro assembler, a linkage editor, a source editor, a mouse based editor, a SHELL development environment, a library of UNIX I/O and utility routines, full access and support of the Macintosh TOOLBOX routines, debugging aides, utilities, make, diff, grep, TTY simulator with upload & download (source supplied), a RAM disk (for 512K Mac), a resource maker, and a **no royalty** license agreement. Programming examples ar included. (Over 600 pages of documentation).

AZTEC C68K-c requires a 128K Macintosh, and two disk drives (frugal developers can make do with one drive). AZTEC C68K supports the 512K Macintosh and hard disks.

| | |
|---|---|
| AZTEC C68K-c (commercial system) | $500 |
| AZTEC C68K-p (personal system) | $199 |
| AZTEC C68K-p to AZTEC C68K-c upgrade | $310 |
| Mac C-tree database | $149 |
| Mac C-tree database with source | $399 |
| Lisa Kit (Pascal to AZTEC C68k object converter) | $ 99 |

### AZTEC C65

**— for the APPLE //**

*"...The AZTEC C-system is one of the finest software packages I have seen..."* NIBBLE review, July 1984.

The only commercial C development system available that runs native on the APPLE II+, IIc, and IIe, the AZTEC C65 development system includes a full floating point C compiler compatible with UNIX C and other MANX AZTEC C compilers, a 6502 relocating assembler, a linkage editor, a library utility, a SHELL development environment, a full screen editor, UNIX I/O and utility subroutines, simple graphics, and screen functions.

| | |
|---|---|
| AZTEC C65 (Apple DOS 3.3) | $199 |
| AZTEC C65/PRO (Apple DOS + ProDos)<br>(call for availability) | $350 |

### AZTEC C II/PRO

**— for CP/M-80**

The first member of the AZTEC C family was the CP/M-80 AZTEC C compiler. It is "the standard" compiler for development on CP/M-80. The system includes the AZTEC C II C compiler, an 8080 assembler, a linkage editor, an object librarian, a full library of UNIX I/O and utility routines, CP/M-80 run time routines, the SMALL library (creates modules less than 3K in size), the fast linker for reduced development times, the ROM library, RMAC and M80 support, library source, support for DRI's SID/ZSID symbolic debugger, and more.

| | |
|---|---|
| AZTEC C II/PRO | $349 |
| AZTEC CII/BAS | $199 |
| C-TREE Database with source | $399 |
| C-TREE Database in AZTEC object form | $149 |

### AZTEC C80

**— for TRSDOS (Radio Shack Model III & 4)**

*"I've had a lot of experience with different C compilers, but the Aztec C80 Compiler and Professional Development System is the best I've seen."* 80-Micro, December, 1984, John B. Harrell III

This sytem has most of the features of AZTEC C II for CP/M. It is perhaps the best software development system for the Radio Shack Model III and IV.

| | |
|---|---|
| AZTEC C80 model 3 (no floating point) | $149 |
| AZTEC C80 model 4 (full) | $199 |
| AZTEC C80/PRO (full for model 3 and 4) | $299 |

To order or for information call:

**800-221-0440**

(201) 530-7997 (NJ and outside U.S.A.). Or write: MANX SOFTWARE SYSTEMS, P.O. Box 55, Shrewsbury, N.J. 07701.

MANX

SHIPPING INFORMATION - Standard U.S. shipment is UPS ground (no fee). In the U.S. one day shipment is $20, two days is $10. Canadian shipment is $10. Two days shipment outside the U.S. is by courier and is freight collect.

TRS 80 RADIO SHACK TRS DOS is a trademark of TANDY.
APPLE DOS MACINTOSH is a trademark of APPLE.

For Technical Support
(Bug Busters) call: 201-530-6557

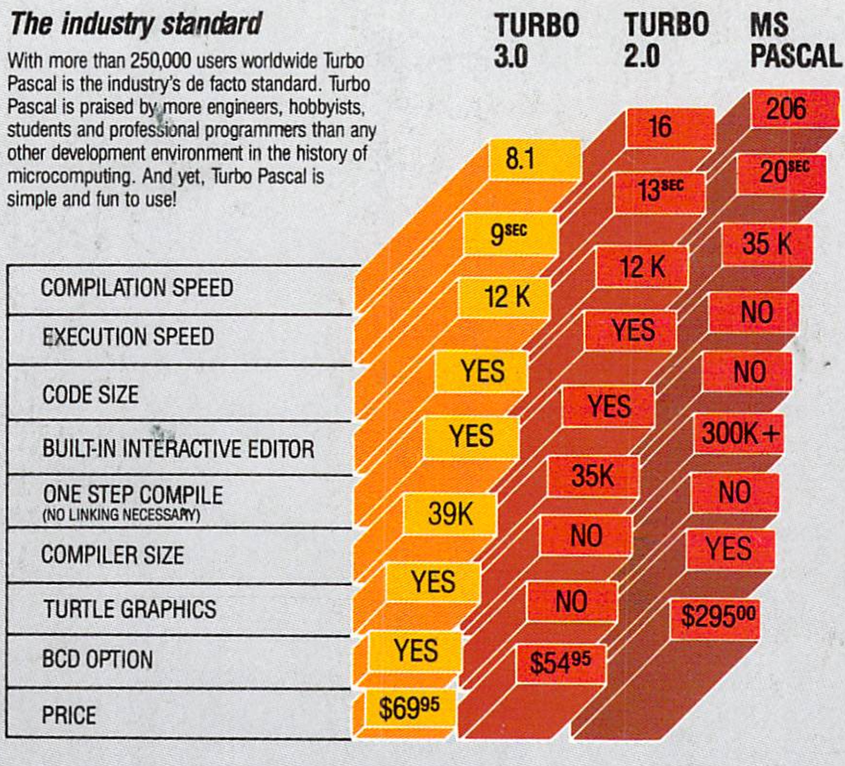
**CIRCLE 69 ON READER SERVICE CARD**

MOST SIGNIFICANT PRODUCT OF THE YEAR - PC WEEK

# They said it couldn't be done. Borland Did It. Turbo Pascal 3.0

### The industry standard

With more than 250,000 users worldwide Turbo Pascal is the industry's de facto standard. Turbo Pascal is praised by more engineers, hobbyists, students and professional programmers than any other development environment in the history of microcomputing. And yet, Turbo Pascal is simple and fun to use!



| | TURBO 3.0 | TURBO 2.0 | MS PASCAL |
|---|---|---|---|
| COMPILATION SPEED | 8.1 | 16 | 206 |
| EXECUTION SPEED | 9 SEC | 13 SEC | 20 SEC |
| CODE SIZE | 12 K | 12 K | 35 K |
| BUILT-IN INTERACTIVE EDITOR | YES | YES | NO |
| ONE STEP COMPILE (NO LINKING NECESSARY) | YES | YES | NO |
| COMPILER SIZE | 39K | 35K | 300K+ |
| TURTLE GRAPHICS | YES | NO | NO |
| BCD OPTION | YES | NO | YES |
| PRICE | $69.95 | $54.95 | $295.00 |

### The best just got better: Introducing Turbo Pascal 3.0

We just added a whole range of exciting new features to Turbo Pascal:

- First, the world's fastest Pascal compiler just got faster. Turbo Pascal 3.0 (16 bit version) compiles twice as fast as Turbo Pascal 2.0! No kidding.
- Then, we totally rewrote the file I/O system, and we also now support I/O redirection.
- For the IBM PC versions, we've even added "turtle graphics" and full tree directory support.
- For all 16 Bit versions, we now offer two additional options: 8087 math coprocessor support for intensive calculations and Binary Coded Decimals (BCD) for business applications.
- And much much more.

### The Critics' Choice.

**Jeff Duntemann, PC Magazine:** *"Language deal of the century . . . Turbo Pascal: It introduces a new programming environment and runs like magic."*

**Dave Garland, Popular Computing:** *"Most Pascal compilers barely fit on a disk, but Turbo Pascal packs an editor, compiler, linker, and run-time library into just 39K bytes of random-access memory."*

**Jerry Pournelle, BYTE:** *"What I think the computer industry is headed for: well documented, standard, plenty of good features, and a reasonable price."*

### Portability.

Turbo Pascal is available today for most computers running PC DOS, MS DOS, CP/M 80 or CP/M 86. A XENIX version of Turbo Pascal will soon be announced, and before the end of the year, Turbo Pascal will be running on most 68000 based microcomputers.

### An Offer You Can't Refuse.

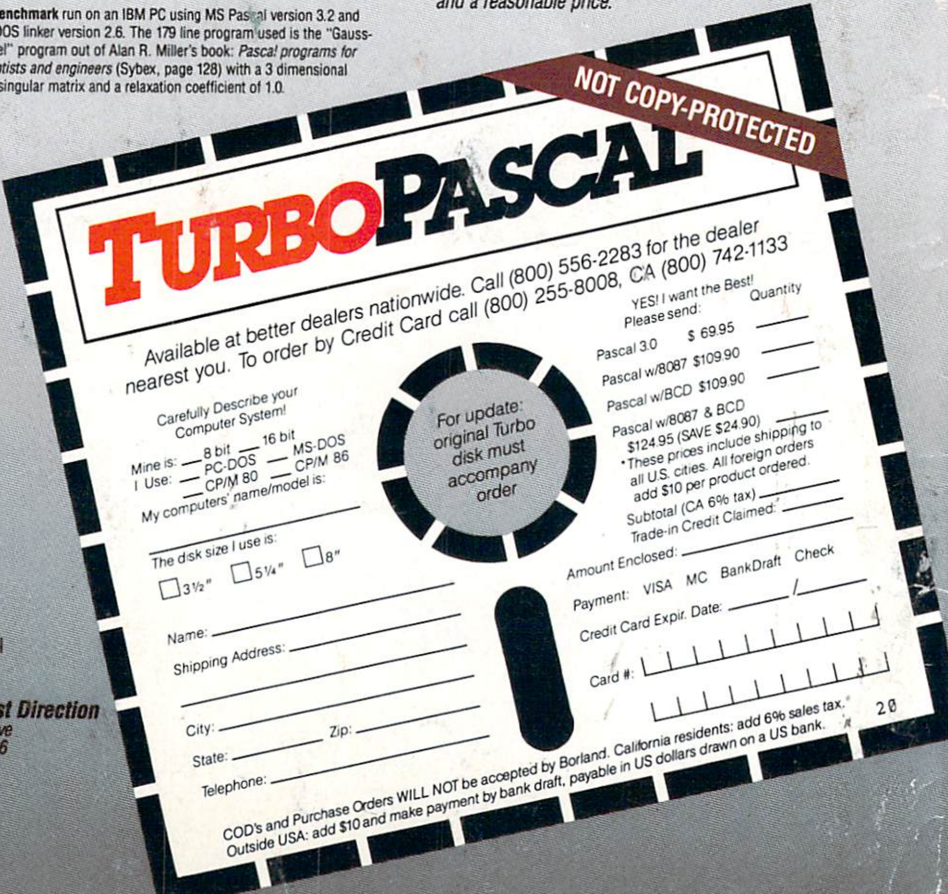
Until June 1st, 1985, you can get Turbo Pascal 3.0 for only $69.95. Turbo Pascal 3.0, equipped with either the BCD or 8087 options, is available for an additional $39.95 or Turbo Pascal 3.0 with both options for only $124.95. As a matter of fact, if you own a 16-Bit computer and are serious about programming, you might as well get both options right away and save almost $25.

### Update policy.

As always, our first commitment is to our customers. You built Borland and we will always honor your support.

So, to make your upgrade to the exciting new version of Turbo Pascal 3.0 easy, we will accept your original Turbo Pascal *disk* (in a bend-proof container) for a trade-in credit of $39.95 and your Turbo87 original *disk* for $59.95. This trade-in credit may only be applied toward the purchase of Turbo Pascal 3.0 and its additional BCD and 8087 options (trade-in offer is only valid directly through Borland and until June 1st, 1985).

(*) **Benchmark** run on an IBM PC using MS Pascal version 3.2 and the DOS linker version 2.6. The 179 line program used is the "Gauss-Seidel" program out of Alan R. Miller's book: *Pascal programs for scientists and engineers* (Sybex, page 128) with a 3 dimensional non-singular matrix and a relaxation coefficient of 1.0.



NOT COPY-PROTECTED

TURBO PASCAL

Available at better dealers nationwide. Call (800) 556-2283 for the dealer nearest you. To order by Credit Card call (800) 255-8008, CA (800) 742-1133

Carefully Describe your Computer System!

Mine is: __ 8 bit __ 16 bit

I Use: __ PC-DOS __ MS-DOS __ CP/M 80 __ CP/M 86

My computers' name/model is: ____

The disk size I use is:

☐ 3½" ☐ 5¼" ☐ 8"

Name: ____

Shipping Address: ____

City: ____ Zip: ____

State: ____

Telephone: ____

For update: original Turbo disk must accompany order

YES! I want the Best! Please send:

| | | Quantity |
|---|---|---|
| Pascal 3.0 | $ 69.95 | ____ |
| Pascal w/8087 | $109.90 | ____ |
| Pascal w/BCD | $109.90 | ____ |
| Pascal w/8087 & BCD | $124.95 (SAVE $24.90) | ____ |

*These prices include shipping to all U.S. cities. All foreign orders add $10 per product ordered.

Subtotal (CA 6% tax) ____

Trade-in Credit Claimed: ____

Amount Enclosed: ____

Payment: VISA MC BankDraft Check

Credit Card Expir. Date: ____/____

Card #: ____

COD's and Purchase Orders WILL NOT be accepted by Borland. California residents: add 6% sales tax. Outside USA: add $10 and make payment by bank draft, payable in US dollars drawn on a US bank.

20

BORLAND INTERNATIONAL

**Software's Newest Direction**
4585 Scotts Valley Drive
Scotts Valley, CA 95066
TELEX 172373

Turbo Pascal is a registered trademark of Borland International, Inc.
PC Week is a trademark of Ziff-Davis Pub. Co.

CIRCLE 9 ON READER SERVICE CARD